VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

KŘÍŽENÍ V KARTÉZSKÉM GENETICKÉM PROGRAMOVÁNÍ

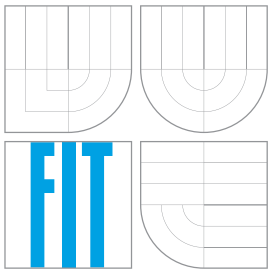
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PETR VÁCHA

BRNO 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

KŘÍŽENÍ V KARTÉZSKÉM GENETICKÉM PROGRAMOVÁNÍ

CROSSOVER IN CARTESIAN GENETIC PROGRAMMING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PETR VÁCHA

VEDOUCÍ PRÁCE
SUPERVISOR

Prof. Ing. LUKÁŠ SEKANINA, Ph.D.

BRNO 2012

Abstrakt

Optimalizace číslicových obvodů se stále těší velké pozornosti nejen u výzkumníků, ale zejména u výrobců čipů. Mezi nové metody umožňující optimalizaci číslicových obvodů patří kartézské genetické programování. Tato diplomová práce se zabývá návrhem a implementací nového operátoru křížení pro kartézské genetické programování. Experimentální vyhodnocení byla provedena v úloze hledání obvodů tříbitové násobičky a pětibitové parity.

Abstract

Optimization of digital circuits still attracts much attention not only of researchers but mainly chip producers. One of new the methods for the optimization of digital circuits is cartesian genetic programming. This Master's thesis describes a new crossover operator and its implementation for cartesian genetic programming. Experimental evaluation was performed in the task of three-bit multiplier and five-bit parity circuit design.

Klíčová slova

křížení, kartézské genetické programovnání, číslicový obvod, násobička

Keywords

crossover, cartesian genetic programming, digital circuit, multiplier

Citace

Petr Vácha: Křížení v kartézském genetickém programování diplomová práce, Brno, FIT VUT v Brně, 2012

Křížení v kartézském genetickém programování

Prohlášení

Prohlašuji, že jsem tento semestrální projekt vypracoval samostatně pod vedením pana Prof. Ing. Lukáše Sekaniny Ph.D. Uvedl jsem všechny literární prameny a publikace, z nichž jsem čerpal.

.....
Petr Vácha
23. května 2012

Poděkování

Rád bych poděkoval svému vedoucímu Prof. Ing. Lukáši Sekaninovi PhD. za vedení práce a rady, které mně poskytl.

Obsah

Kapitola 1

Úvod

Evoluční algoritmy v současnosti představují zastřešující název pro třídu moderních matematických postupů využívajících modely evolučních procesů přírody. Do této skupiny řadíme mimo jiné evoluční programování, evoluční strategie, genetické algoritmy a genetické programování. Jedná se o populární oblast umělé inteligence, která nachází velké uplatnění v oblasti optimalizace.

Kde selhávají konvenční inženýrské přístupy, tam je šance uplatnit inovativní řešení evolučních algoritmů. Jednou z těchto oblastí je i návrh číslicových kombinačních obvodů. Pro dosažení levnějšího a rychlejšího řešení obvodu se používá tzv. kartézské genetické programování (CGP[1]), což je varianta genetického programování. CGP se tak stává metodou, která je výkonným pomocníkem inženýra [8].

Práce zpracovává studii zabývající se současným stavem CGP, popisuje genetické operátory a aplikace v oblasti číslicových obvodů. Primárně se tato diplomová práce zabývá rozšířením CGP algoritmu o operátor křížení. Toto je velmi zajímavá úloha, protože se toto doposud nikomu nepodařilo uspokojivě vyřešit.

Struktura textu této práce je rozdělena do osmi částí. V kapitole 2 jsou popsány genetické operátory. Kapitola 3 se zabývá popisem principu CGP, návrhem obvodů pomocí CGP a seznamuje čtenáře s nástrojem pro CGP. Výběrem vhodného problému pro experimenty se zabývá 4. kapitola. V této kapitole je možné se seznámit s používaným statistickým vyhodnocením. Následující kapitola popisuje navržené neinformované operátory křížení. V 6. kapitole je navrženo nové sémantické křížení pro CGP. Závěrečná kapitola vyhodnocuje provedené experimenty s navrženými operátory křížení.

[1] Cartesian Genetic Programming

Kapitola 2

Evoluční algoritmus a genetické operátory

Tato kapitola popisuje princip evoločního algoritmu a způsob aplikace genetických operátorů. Genetické operátory jsou používaný v průběhu evolučního procesu pro trasformaci původní populace na populaci novou. Mezi hlavní genetické operátory řadíme operátory křížení a mutace. Připomeňme si, že v genetickém programování pracujeme se stromově orientovanou strukturou, a že za hlavní operátor genetického programování je považováno křížení. Ale taktéž, ač v menším měřítku, je využíván i operátor mutace [8]. V klasické verzi CGP představené v roce 1999 Julianem Millerem se nepoužívají žádné varianty křížení [5]. V posledních letech se objevují modifikace CGP, které se snaží využít potenciálu křížení. V některých případech dávají dobré výsledky. Například pro řešení problémů symbolické regrese navržených Johnem Kozou vedou k více než 50% redukci počtu generací nutných k nalezení výsledku pro některé případy [1]. Pro nalezení obrazových filtrů se ukazuje společně s mutací i jednobodové křížení jako vhodný operátor [10]. K nalezení kvalitního obrazového filtru je potřeba jen přibližně 30 tisíc generací. Naproti tomu, k nalezení dobré tříbitové násobičky, je potřeba asi 5 milionů generací. Ale je nutné podotknout, že ačkoliv k nalezení obrazového filtru není potřeba tolik generací, tak samotné nalezení filtru zabere z celkového časového hlediska delší dobu než nalezení násobičky [10].

2.1 Evoluční algoritmus

Evoluční algorimus používá populaci kandidátních řešení k paralelnímu prohledávání stavového prostoru. Pro vytváření nových kandidátních řešení se používají biologií inspirované operátory.

V prvním kroku výpočtu je vytvořena počáteční populace, která obsahuje předem zvolený počet kandidátních jedinců. Počáteční populace je buď zvolena zcela náhodně, nebo pomocí vhodné heuristiky, která např. může využívat již existující řešení problému. Kandidátní řešení se mohou reprezentovat jako řetězce bitů, řetězece integerů nebo jako stromé struktury (jedná-li se o genetické programování). V každém kroku algoritmu, který nazýváme generace, jsou všechna kandidátní řešení nejdříve ohodnocena pomocí tzv. *fitness funkce*. Lepší jedinci jsou hodnoceni vyššími číselnými hodnotami. Každá nová populace vznikne tak, že se ze stávající množiny určí jedinci, kteří utvoří množinu rodičů. O výběr jedinců z množiny rodičů se stará tzv. *selekční algoritmus*. Pomocí genetických operátorů nad množinou rodičů vznikne nová množina potomků. Nejznámějšími genetickými operá-

tory jsou křížení a mutace. Z této množiny potomků a rodičů se vyberou jedinci pro další populaci.

Pokud je nová populace utvářena pouze pomocí nově vzniklých jedinců, tj. úplnou obnovou (vymírání rodičů), hovoří se o *generační variantě* evolučního algoritmu. O *překrývání populací* mluvíme tedy, pokud nová generace obsahuje jak původní jedince tak i nově vzniklé. Jedná se tedy o částečnou obnovu populace (steady state). Dalším důležitým pojmem v evolučních algoritmech je tzv. *elitismus*, který zaručuje, že nejlepší jedinec z předchozí populace se vždy dostane do nové populace. Algoritmus je ukončen při nalezení dostatečně kvalitního jedince nebo po vyčerpání zadaného počtu generací [8]. Princip evolučního algoritmu je shrnutý v níže uvedeném algoritmu 1 (algoritmus byl převzatý z [8]).

```
ALGORITMUS 1:
evolucni_algoritmus() {
    t=0
    P(t) := vytvor_pocatecni_populaci();
    ohodnot(P(t));
    while (ukoncovaci_podminka == FALSE) {
        Q(t) := vyber_rodice(P(t));
        Q'(t) := vytvor_nove_jedince(Q(t));
        ohodnot(Q'(t));
        P(t+1) := vyber_jedince_do_nove_populace(P(t), Q'(t));
        t := t + 1;
    }
}
```

2.1.1 Selekční algoritmy

Společným rysem selekčních algoritmů je, že se snaží vybírat jedince s vyšší hodnotou fitness a zároveň zajišťovat dostatečnou různorodost pro nové populace. Mezi selekční algoritmy patří tzv. *proporcionální selekce*. Vzorec 2.1 definuje tento selekční mechanizmus jako pravděpodobnost výběru pro i-tého jedince. Hodnota f_i je fitness i-tého jedince a N je velikost populace. Pravděpodobnost výběru je tedy přímo úměrná absolutní hodnotě fitness jedince. Výběr je proveden pomocí tzv. kola štěstí (rulety), kde takto ohodonocení jedinci vymezují svou pravděpodobností p_i pomyslné úseky na ruletě. Výběr probíhá náhodným vygenerováním čísla v rozsahu $< 0, 1 >$. Toto číslo potom určuje vybraný úsek na kole štěstí. Jedinci s větší pravděpodobností mají na intervalu větší část než jedinci s menší pravděpodobností, tím se zachovává proporcionalita. Pro výběr K jedinců je proces generování náhodného čísla v intervalu $< 0, 1 >$ K-krát opakován.

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \tag{2.1}$$

Variantou této metody je *výběr podle pořadí*. Jako první se seřadí kandidátní jedinci podle fitness hodnot. Pravděpodobnost výběru je potom úměrná pořadí jedince a ne hodnotě fitness. Díky této variantě se dá předcházet zdegenerování populace.

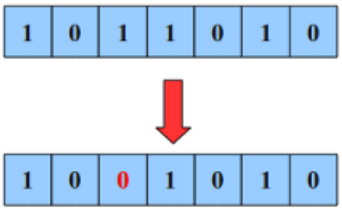
Další metodou je tzv. *turnajová selekce*, která porovnává náhodně vybraných s jedinců. Jedinec s nejvyšší fitness ve skupině představuje vítěze turnaje a postupuje do dalšího kroku algoritmu. Tento proces je opakován tolikrát, kolik jedinců má být vybráno.

Deterministická selekce je typem selekce, která deterministicky vybírá K jedinců s nejvyšší hodnotou fitness [8].

2.2 Operátor mutace

Operátor mutace očekává na vstupu jednoho jedince. Výsledkem aplikace tohoto operátoru je jeden nový jedinec s určitým počtem modifikací v chromozomu. V případě binárního zakódování je invertován určitý počet bitů (genů) s pravděpodobností p_m. V GA[1] je tato pravděpodobnost velmi malá, volí se hodnoty okolo 0,1 %. Tato malá pravděpodobnost potom umožňuje ovlivňovat kandidátní řešení postupně po malých krocích [8].

Pokud je reprezentace chromozomu celočíselná, tak je nutné zajistit, aby výsledný gen byl opět platný resp. legální pro danou instanci zakódování. V případě standardní varianty CGP[2] operátor mutace vybere gen a náhodně vygeneruje jeho novou hodnotu. Parametrem operace je počet takto modifikovaných genů [8]. Typická ukázka realizace operátoru mutace je na obrázku 2.1. Existují i další varianty mutace, např. pro reálně kódované chromozomy.



Obrázek 2.1: Aplikace operátoru mutace. Horní chromozom zobrazuje rodiče, spodní zobrazuje kopii rodiče s mutací jednoho genu (invertování bitu).

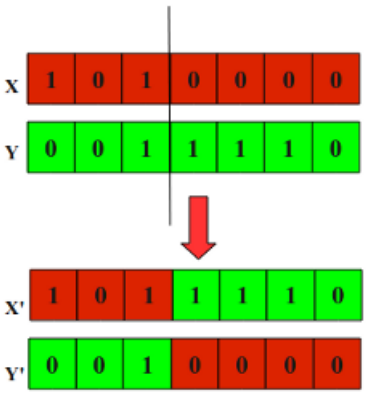
2.3 Operátor křížení

Operátor křížení obvykle očekává na vstupu dva jedince. Z těchto dvou jedinců vytváří jednoho, nebo dva nové. Nový jedinec vznikne opět v případě binární reprezentace kombinací n prvků z prvního rodiče a m prvků z druhého rodiče. Počet prvků nového jedince je shodný s délkou jednotlivých rodičů. V GA je křížení v rodičovské populaci aplikováno s pravděpodobností p_c, která se obvykle pohybuje kolem hodnoty $p_c = 0{,}7$. To znamená, že 70 % jedinců nově vzniklé populace vznikne křížením rodičů, zbytek tvoří kopie, popřípadě mutované kopie [8]. Rozlišujeme následující základní varianty křížení.

2.3.1 Jednobodové křížení

U jednobodového křížení je náhodně vybrán bod křížení, v němž dojde k prohození genů, které se nacházejí za bodem křížení [8]. Obrázek 2.2 nám ukazuje aplikaci jednobodového křížení za vzniku dvou potomků.

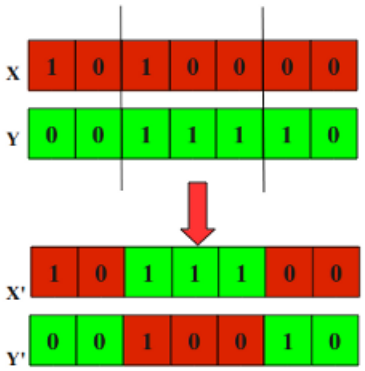
[1] Genetický algoritmus
[2] Podrobný popis této varianty je uvedený v kapitole Princip CGP (3.1)



Obrázek 2.2: Aplikace operátoru jednobodového křížení. V horní části vidíme rodiče X a rodiče Y. Svislá čára procházející rodiči znázorňuje zvolený bod křížení. Po aplikaci operátoru vznikají dva noví potomci, potomek X' a potomek Y'.

2.3.2 Vícebodové křížení

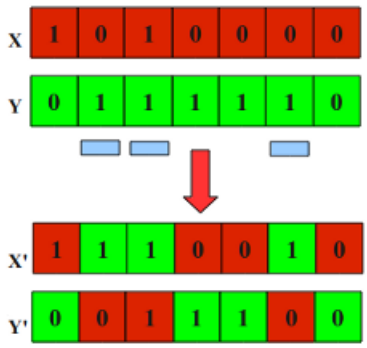
Ve vícebodovém křížení se definuje větší počet bodů pro křížení. To má za důsledek, že genetický materiál je tak mezi rodiči daleko více promíchaný [8]. Na obrázku 2.3 vidíme aplikaci tohoto vícebodového křížení.



Obrázek 2.3: Aplikace operátoru vícebodového křížení. V horní části vidíme rodiče X a rodiče Y. Svislé čáry procházející rodiči znázorňují zvolené body křížení. Po aplikaci operátoru vznikají dva noví potomci, potomek X' a potomek Y'.

2.3.3 Uniformní křížení

Případ uniformního křížení je charakteristický tím, že pro každý gen je zvlášť rozhodnuto, zda-li dojde k jeho prohození mezi rodiči [8]. Obrázek 2.4 nám ukazuje princip uniformního křížení.



Obrázek 2.4: Aplikace operátoru uniformního křížení. V horní části vidíme rodiče X a rodiče Y. Modré značení říká, které geny budou vyměněny. Po aplikaci operátoru vznikají dva noví potomci, potomek X' a potomek Y'.

2.3.4 Křížení ve stromovém genetickém programování

Jedna z variant operátoru křížení ve stromově orientovaném GP[3] kombinuje genetický materiál dvou rodičů prohozením části jednoho rodiče s částí druhého rodiče. Po výběru dvou rodičů se náhodně v každém rodiči označí uzel s příslušným podstromem. Nakonec se prohodí označené podstromy mezi oběma rodiči. Výsledkem jsou dva potomci [8]. Obrázek 2.5 nám ukazuje princip křížení ve stromovém GP.

2.3.5 Sémanticky založené křížení

Předchozí varianty křížení nebraly v potaz sémantický význam zakódování jedince. Dvě syntakticky stejné struktury musejí mít stejný sémantický význam. Naproti tomu ale dvě sémanticky stejné struktury nemusí mít stejnou syntaktickou stavbu. Pro představu si vezměme dva jednoduché vztahy,

$$a = 1; b = a + a + a, \tag{2.2}$$

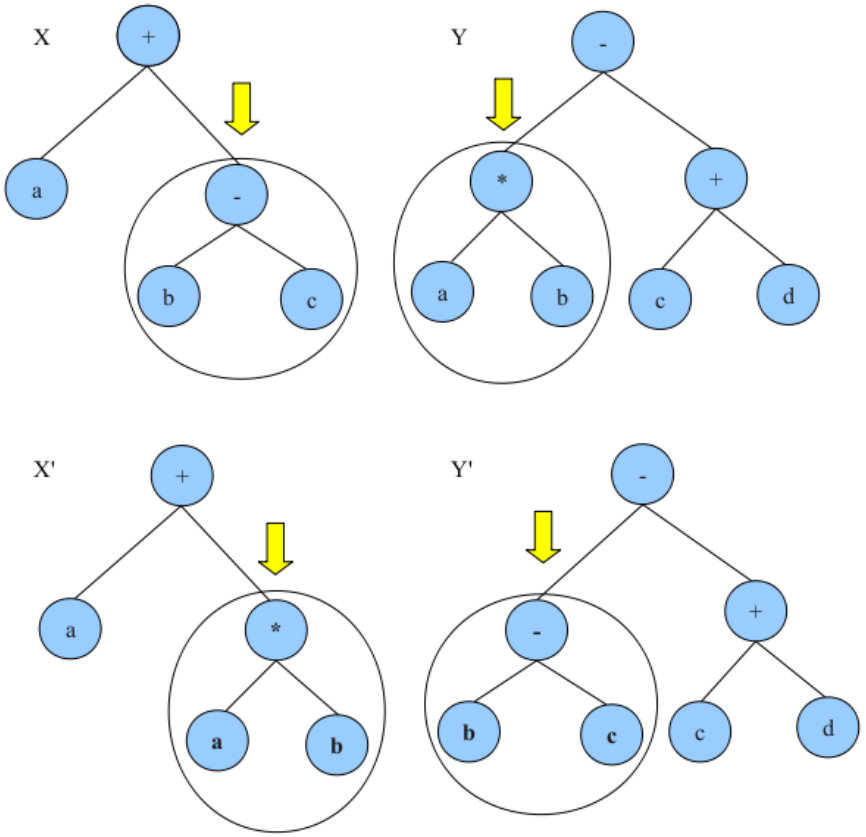
$$a = 1; b = 3 * a \tag{2.3}$$

.

Sémantický význam vztahu 2.2, význam hodnoty b, je stejný jako v 2.3. V takovém případě hovoříme o sémantické ekvivalenci [4].

V [11] je uveden popis pro vyšetření vzorkovací sémantické vzdálenosti[5] pro GP. Nejprve se na zvolených stromových strukturách T_1 a T_2 provede tzv. *sémantické vzorkování*[6]. Tyto jednotlivé (pod)stromy se ohodnotí pro zvolenou množinou vstupních hodnot P, která obsahují N hodnot. Například pro $N = 3$, $P = \{0, 0.5, 1\}$ a kandidátní podstrom T_1, uvedený na obr. 2.6, mohou výsledné ohodnocené množiny nabývat těchto hodnot $St_1 = \{sin(1) - 0, sin(1) - 0.5, sin(1) - 1\} = \{0.84, 0.34, -0.16\}$) Výpočet SSD metriky je uvedený

[3] genetické programování
[4] Semantic Equivalent
[5] Sampling Semantics Distance (SSD)
[6] Sampling Semantics (SS)



Obrázek 2.5: Jedna z variant křížení v genetickém programování. Z rodičů X a Y vzniknou potomci X' a Y'.

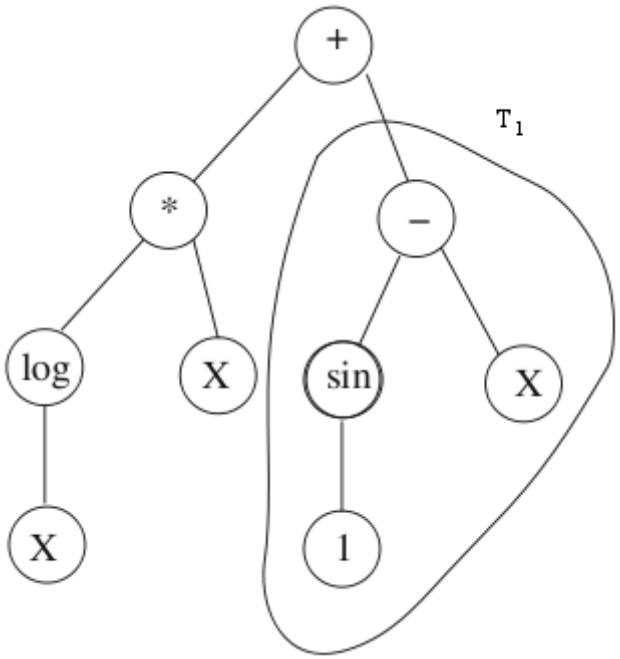
ve vztahu 2.4. Hodnoty p_i jsou ohodnocení pro vstupní hodnoty P pro podstrom T_1 a q_2 pro podstrom T_2.

$$SSD(St_1, St_2) = (|p_1 - q_1| + |p_2 - q_2| + \cdots + |p_N - q_N|)/N \tag{2.4}$$

Pokud je $SSD(St_1, St_2) < \epsilon$, jedná se o sémantickou ekvivalenci. Hodnota ϵ se volí dostatečné malá a nazývá se *sémantická citlivost*.

Sémanticky založené křížení tak bere na zřetel význam jednotlivých částí jedince. SAC[7] je jedna z variant sémantického křížení v GP, která zabraňuje výběru podstromů rodičů, které si jsou sémanticky ekvivalentní. SSC[8] je metoda, která rozšiřuje SAC dvěma způsoby. Zaprvé, podstromy jsou vybrány ke křížení, pokud jsou si sémanticky podobné. Zadruhé, jelikož je nalezení sémanticky podobných podstromů mnohem náročnější, než pouhé vyhnutí se sémanticky ekvivalentním podstromům, je zde větší počet opakování hledání takových podstromů. V porovnání se širokou škálou křížení na různých problémech je SSC prokazatelně lepší než standardní křížení nebo SAC [11]. SSC je popsáno pseudokódem jako algoritmus 2.

[7] Semantics aware crossover
[8] Semantic similarity-based crossover



Obrázek 2.6: Příklad podstromu T_1 v GP.

```
ALGORITMUS 2:
SSC() {
    P1 := vyber_prvniho_rodice();
    P2 := vyber_druheho_rodice();
    Pocet := 0;
    while (pocet < MAX_POCET_POKUSU) {
        podstrom1 = nahodne_zvol_podstrom(P1);
        podstrom2 = nahodne_zvol_podstrom(P2);
        nahodne_vygeneruj_vstupni_hodnoty_pro_dany_problem();
        if(SSD(podstrom1, podstrom2) < epsilon) {
            proved_krizeni();
            pridej_potomky_do_nove_populace();
            return true;
        } else {
         pocet := pocet + 1;
        }
    }
    if (pocet == MAX_POCET_POKUSU) {
        podstrom1 = nahodne_zvol_podstrom(P1);
        podstrom2 = nahodne_zvol_podstrom(P2);
        proved_krizeni();
    }
}
```

2.3.6 Křížení v CGP

Problematikou CGP se zabývá až následující kapitola. Nicméně již zmíníme operátor křížení vyvinutým pro CGP. Jedna z uspokojivě pracujících metod křížení v CGP je do značné míry inspirována křížením v GA, který používá v reprezentaci reálné hodnoty. V této metodě je genotyp CGP překódován z celočíselné reprezentace na reprezentaci s reálnými hodnotami. Každá tato hodnota odpovídá jednomu genu v CGP genotypu. Hodnoty nového zakódování leží v intervalu [0, 1]. Křížení potom probíhá stejně jako v GA, který využívá chromozomy s reálnými hodnotami. Více o této technice křížení je v [1].

Kapitola 3

CGP a návrh obvodů

Tato kapitola seznamuje čtenáře se základní variantou CGP. První podkapitola popisuje způsob zakódování a konstrukci fitness funkce pro evoluční návrh číslicových kombinačních obvodů. Druhá podkapitola se věnuje popisu implementace CGP, která byla vyvinuta na FIT VUT v Brně[1] [16].

3.1 Princip CGP

CGP je variantou genetického programování, u které jsou kandidátní řešení reprezentována pomocí obecných (v případě kombinačních obvodů acyklických) orientovaných grafů [8]. Kartézské genetické programování má přívlastek 'kartézské', protože reprezentuje program pomocí dvoudimenzionální mřížky uzlů [6]. V případě návrhu číslicových kombinačních obvodů hovoříme o rekonfigurovatelném obvodu, který je modelován jako pole elementů, představující uzly grafu, o velikosti n_c (počet sloupců) $\times$ n_r (počet řádků). Jednotlivé uzly pak reprezentují právě jednu funkci s n_n argumenty, která je vybrána z množiny dostupných funkcí Γ. Počet těchto funkcí budeme značit n_f. Počet vstupů n_i a výstupů n_o obvodu je pevně určený na začátku evoluce. Vstupy uzlu, který se nachází v i-tém sloupci, mohou být připojeny buď na primární vstupy obvodu, nebo na výstupy uzlů umístěných až v L předchozích sloupcích. To znamená, že vstupy uzlu v i-tém sloupci nesmějí být připojeny na výstupy uzlů ve stejném nebo následujícím sloupci. Parametr L je tzv. L-*back* parametr, který určuje míru propojitelnosti obvodu. Pro $L = 1$ je propojitelnost minimální - mohou se propojovat pouze sousední uzly. Relativně největší propojitelnosti se dá dosáhnout pokud bude $L = n_c$ [8]. Na obrázku 3.2 můžeme vidět instanci CGP.

Vstupy obvodu jsou indexovány z intervalu 0, ..., n_i - 1. Rovněž i výstupy uzlů jsou indexovány po sloupcích s počáteční hodnotou n_i pro nejlevější horní uzel. Uzel obvodu je reprezentován $n_n + 1$ celočíselnými hodnotami. Prvních n_n hodnot určuje indexy uzlů, na které bude připojen vstup daného uzlu. Poslední hodnota označuje logickou funkci uzlu. Posledních n_o hodnot chromozomu určuje indexy uzlů, ke kterým budou připojeny výstupy obvodu. Důležité je si uvědomit, že takto reprezentované kódování chromozomu má neměnou konstatní velikost, ale velikost zakódovaného obvodu (fenotypu) je variabilní. Obrázek 3.3 nám ukazuje fenotyp instance z obrázku 3.2.

Algoritmus CGP pro prohledávání takto daného stavového prostoru je podobný variantě evolučních strategiích, která se označuje ES $(1 + \lambda)$. Tato varianta obsahuje $1 + \lambda$ jedinců (chromozomů) v populaci. Novou populaci tak tvoří nejlepší jedinec z předchozí populace

[1] Fakulta informačních technologií, Vysoké učení technické v Brně

a λ jeho mutantů. Celková velikost populace se na základě zkušeností volí kolem 5 [15]. Pro ohodnocení chromozomu, resp. pro ohodnocení výsledného fenotypu zapojení, se používá tzv. fitness funkce.

Fitness funkce zohledňuje v první řadě správnost výsledného řešení. V případě ohodnocování evolučního návrhu a optimalizace kombinačních obvodů je zapotřebí sestavit pravdivostní tabulku pro daný problém. Kandidátní řešení otestujeme pro každou možnou kombinaci vstupů a výsledek porovnáme s požadovaným výstupem. Výsledná fitness je pak rovna počtu správných bitů ve výstupním vektoru, tzn. bitů shodujících se s požadovaným výstupem. Pro dosažení optimalizace počtu hradel se používá modifikovaná fitness funkce. Ta se aplikuje až po dosažení maximální hodnoty fitness (tj. $f_m = n_o 2^{n_i}$). Modifikovaná fitness má tvar:

$$f = f_m + n_r n_c - g, \tag{3.1}$$

kde g označuje počet použitých hradel v kandidátním obvodu. Tato funkce dává vyšší prioritu obvodům, které zcela realizují požadovanou funkčnost a obsahují co nejméně hradel.

Standardní varianta CGP používá pouze operátor mutace. Tento operátor mutace náhodně vybere gen a náhodně vygeneruje jeho novou hodnotu. Parametrem mutace je počet takto změněných genů, který značíme μ_g. Mutace má vliv na změnu funkce uzlu, připojení vstupního signálu k uzlu nebo připojení primárního výstupu obvodu.

V souvislostí s fitness funkcí je důležité také zmínit pojem neutrální mutace. Neutrální mutace nemá vliv na aktuální fitness hodnocení. Pokud má mutace vliv na hodnocení, nazýváme ji adaptivní. Přítomnost těchto neutrálních mutací je velmi příznivým jevem. Některé obvody by bez něj nemohly být vůbec nalezeny [6].

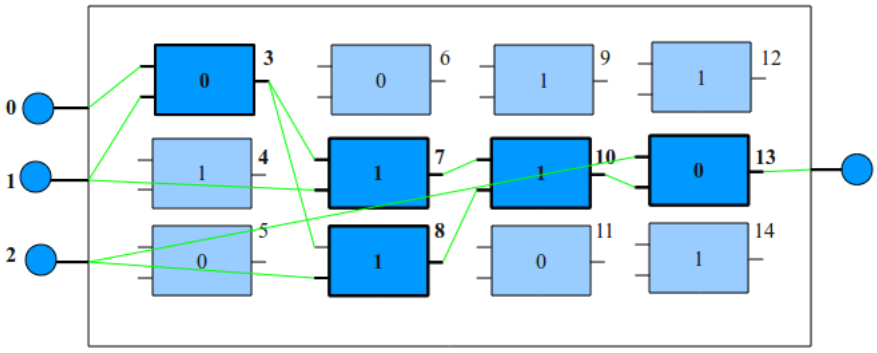
Nová populace se vytváří ze staré populace. Nejprve se vybere jedinec s nejlepší hodnotou fitness a vloží se do nové populace spolu se svými λ mutanty. Pro zajištění genetické diverzity se v případě, že existuje více "nejlepších"řešení se stejnou hodnotou fitness, použije se jako rodič ten, který nebyl rodičem v předchozí generaci 3.1. Počáteční generace se většinou generuje náhodně. Evoluce končí nalezením kvalitního řešení, nebo vyčerpáním povoleného počtu generací [8].

Obrázek 3.1: ES $(1 + \lambda)$: Pokud existuje více nejlepších jedinců se stejnou fitness hodnotou, použije se jako rodič ten, který nebyl rodičem v předchozí generaci.

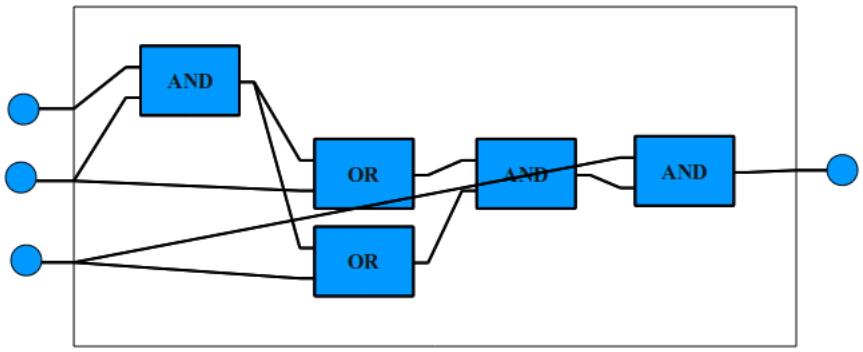
CGP algoritmus uvedený v [8]:

1. Vygenerování $1 + \lambda$ náhodných jedinců pro inicializaci populace.

2. Ohodnocení všech jedinců populace pomocí fitness funkce.
3. Nalezení nejlépe ohodnoceného jedince (kontrola duplicity s předchozí generací).
4. Vygenerování λ potomků z nejlepšího jedince pomocí mutace.
5. Nejlepší jedinec společně s jeho λ potomky tvoří novou populaci.
6. Pokud není splněna ukončující podmínka, pokračuje se krokem 2.



Obrázek 3.2: Příklad instance CGP s parametry $n_r = 3$, $n_c = 4$, $n_i = 3$, $n_o = 1$, $n_n = 2$, $n_f = 2$, $L = n_c$, $\Gamma = \{\text{AND } (0), \text{OR } (1)\}$. Uzly 4, 5, 6, 9, 11, 12 a 14 nejsou ve výsledku použity. Chromozom: 0,1,0, 1,0,1, 1,2,0, 0,1,0, 3,1,1, 3,2,1, 0,1,1, 7,8,1, 2,1,0, 0,3,1, 2,10,0, 10,11,1, 13. Poslední hodnota chromozomu určuje zapojení výstupu.



Obrázek 3.3: Fenotyp zapojení pro předchozí konfiguraci chromozomu.

3.2 Implementace CGP

V knize [6] najdeme seznam existujících implementací CGP. Jako vhodný nástroj pro CGP se zdá být volně dostupná sada nástrojů Tools4CGP[2].

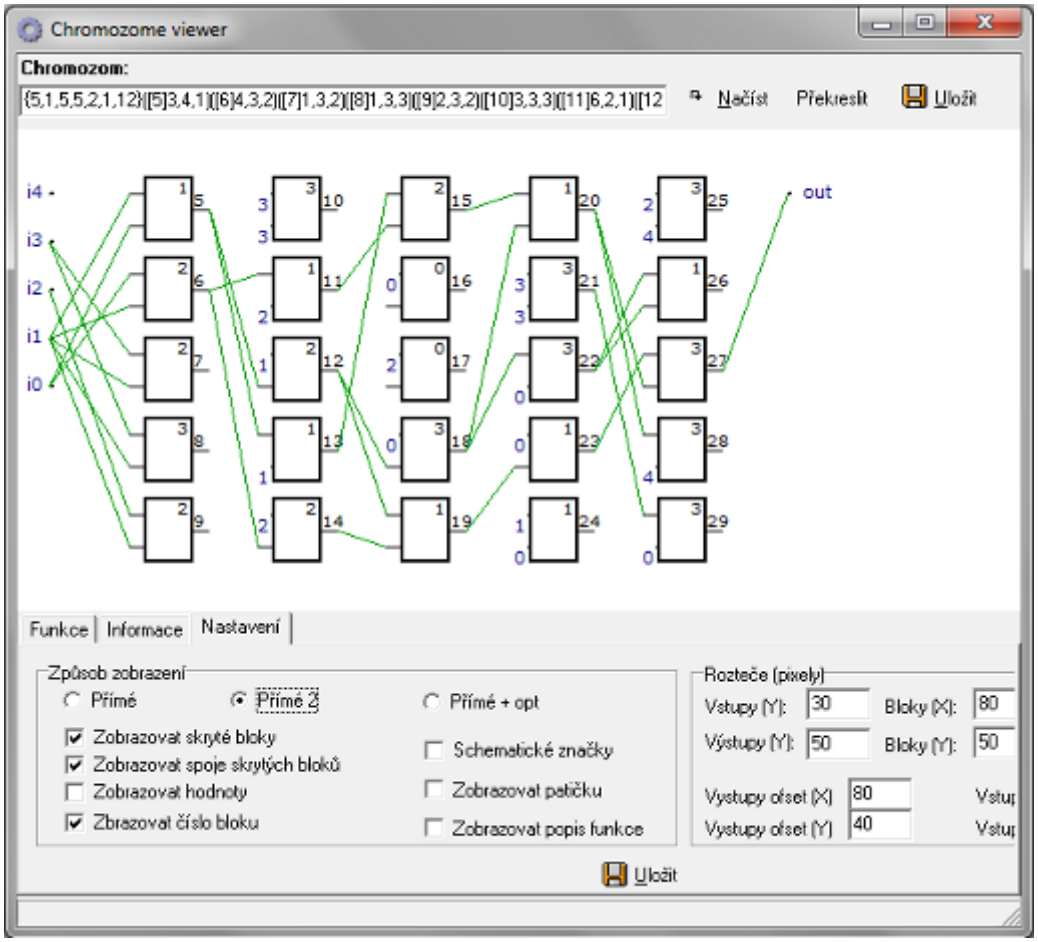
[2] http://www.fit.vutbr.cz/research/view_product.php?id=61&notitle=1

3.2.1 Popis sady Tools4CGP

Projekt je kompletně napsaný v jazyce C++. Skládá se ze tří samostatných programů: Tab2h, CGP, Chromozome viewer. Program CGP obsahuje samotnou implementaci CGP. Program tab2h převádí zadanou pravdivostní tabulku na .h soubor, který se použije pro evoluci. Cgpviewer je prohlížeč chromozomů, který umožňuje zobrazit chromozomy CGP, simulovat je a exportovat do různých formátů (VHDL, bmp). Program cgpviewer je spustitelný pouze pod operačním systémem Windows. Ostatní dva zmíněné programy jsou nezávislé na platformě.

Nástroj CGP má implementovanou tzv. *paralelní simulaci*, která využívá existence bitových operátorů v programovacím jazyce C. Díky paralelní simulaci získáváme v závislosti na zvolené architektuře až několikanásobné zrychlení. Více informací o paralelní simulaci je v [8].

Výstupem úspěšného běhu programu CGP (nalezení řešení) je soubor .chr, který obsahuje nejlepší nalezené řešení. Tento soubor se dá prohlížet cgpviewer programem. Na obrázku 3.4 je vidět jeho použití. Výhodou tohoto programu je možnost přesouvání hradel a snadný export schématu do obrázku. Výstupem tohoto programu je také .xls soubor, do kterého je generován průběh evoluce.



Obrázek 3.4: Náhled na program cgpviewer.

Parametry programu CGP:

Přehled parametrů je popsaný v technické dokumentaci [14]:

POPULACE_MAX ... počet jedinců populace
MUTACE_MAX ... maximální počet genů mutovaných během jedné mutace, podle tohoto čísla se náhodně vygeneruje, kolik se má zmutovat genů, a poté se náhodně generuje, které geny se mají mutovat
PARAM_M ... počet sloupců matice
PARAM_N ... počet řádků matice
L_BACK ... l-back parametr
PARAM_GENERATIONS ... počet generací, po kterých se má evoluce ukončit
PARAM_RUNS ... počet běhů evoluce – kolikrát se má znovu spustit
FUNCTIONS ... počet použitých funkcí

Naprogramované funkce, které mohou realizovat jednotlivé bloky
0 výstup = vstup1
1 výstup = vstup1 **and** vstup2
2 výstup = vstup1 **or** vstup2
3 výstup = vstup1 **xor** vstup2
4 výstup = **not** vstup1
5 výstup = **not** vstup2
6 výstup = vstup1 **and not** vstup2
7 výstup = **not** (vstup1 **and** vstup2)
8 výstup = **not** (vstup1 **or** vstup2)
PARAM_IN ... počet vstupů komb. obvodu
PARAM_OUT ... počet výstupů komb. obvodu
DATASIZE ... počet integerů, které se použijí jako testovací vektory

3.3 Aplikace CGP

CGP bylo aplikováno v mnoha oblastech. Například jde o návrh filtrů, medicínské problémy, evoluční umění, návrh číslicových obvodů, dekompozice problémů, symbolická regrese, návrh kontrolerů pro roboty a jiné.

Nevýhodou evolučního návrhu obvodu pomocí CGP je, že nelze aplikovat pro obvod s mnoha vstupy. Potenciální řešení nabízí formální verifikace kandidátních řešení [12].

Kapitola 4

Experimenty a statistické vyhodnocení

Tato kapitola se zabývá výběrem vhodných problémů pro ověření výkonnostní kvality CGP a jeho variant. Pro statistické zhodnocení výsledků experimentů jsou zde prezentovány nejčastěji používané metody.

4.1 Výběr vhodného problému

Pomocí CGP lze reprezentovat, potažmo řešit, různé problémy. Například to může být řešení problému symbolické regrese, návrh obrazů nebo návrh elektronických obvodů. Dále se zaměříme na aplikaci v oblasti návrhu číslicových kombinačních obvodů.

Návrh jednoduchých kombinačních obvodů byl první aplikací CGP, která dávala nekonvenční, ale použitelné řešení. CGP tak umožnilo nalezení efektivnějších řešení, než konvenční metody. V oblasti návrhu kombinačních obvodů se typicky jedná o redukci počtu hradel nebo zmenšení zpoždění obvodu T.

Mezi úlohy pro ověření CGP na úrovni hradel patří problém optimalizace sčítačky, paritní funkce a násobičky. Se současným stavem násobiček nás seznamuje tabulka 4.1. Z této tabulky je zřejmé, že CGP přináší velkou úsporu hradel oproti konvenčním řešením. *Nejlepší CGP* výsledky z tabulky jsou dosaženy za použití $\Gamma = \{xANDy, xXORy, (not)xANDy\}$. Hradlo $(not)xANDy$ je zde uvažováno jako jedno, i když se ve skutečnosti jedná o hradla dvě. Přepočet, kdy je toto hradlo započítáno jako hradla dvě, je ve sloupci *Přepočítané CGP*. Pro dosažení těchto výsledků bylo použito následující nastavení parametrů evoluce: $L = n_c$, $\lambda = 4$, $\mu_g = 3$ [9].

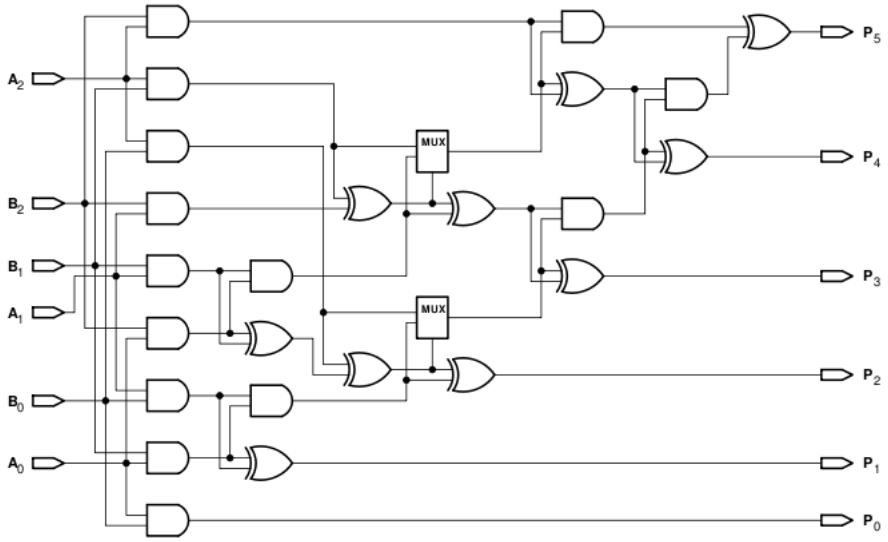
Násobička	Nejlepší konv.	Nejlepší CGP	Přep. CGP	$n_c \times n_r$	Max. generací
2b × 2b	8	7	9	1 × 7	10k
3b × 2b	17	13	14	1 × 17	200k
3b × 3b	30	23	25	1 × 35	20M
4b × 3b	47	37	44	1 × 56	200M
4b × 4b	64	57	67	1 × 67	700M

Tabulka 4.1: Přehled násobiček za použití dvouvstupých hradel. Převzato z [9].

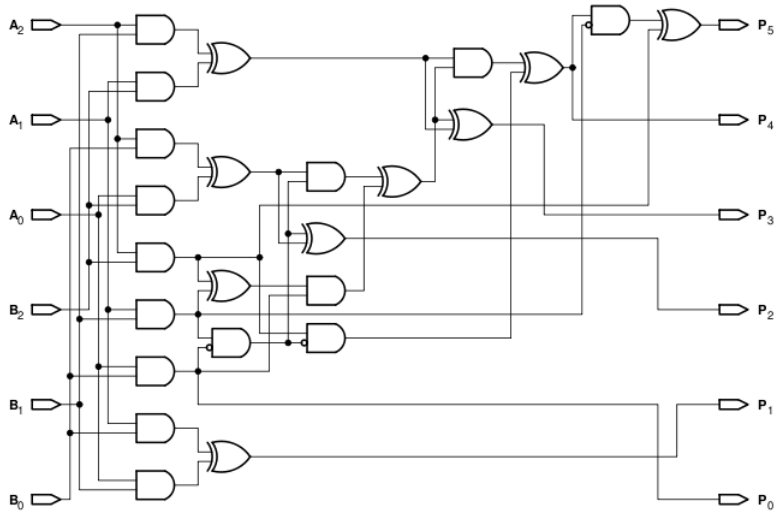
4.1.1 Tříbitová násobička

Tříbitová násobička se začíná řadit ke složitějším kombinačním obvodům, které byly navrženy pomocí CGP. Má dva tříbitové vstupy (celkem tedy šest vstupů) a šest výstupů (f_m = 384). Na obrázku 4.1 je nejlepší konvenční řešení tříbitové násobičky s dvouvstupovými hradly a multiplexory, které obsahuje 26 hradel [2].

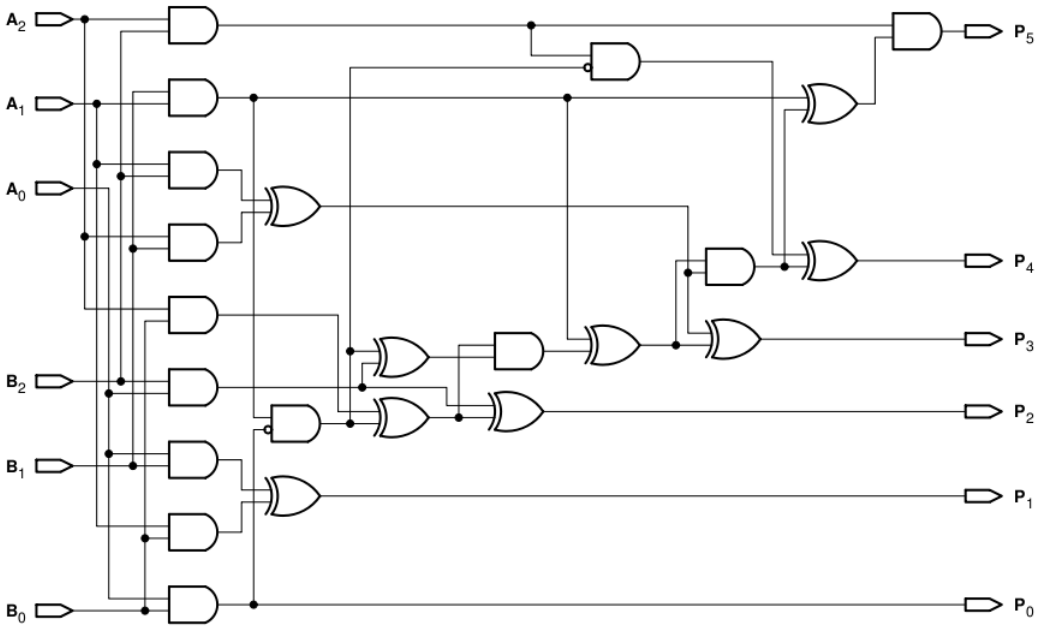
Lepší řešení z obrázku 4.2 s 24 hradly bylo nalezeno ze 100 evolučních běhů, kdy každý obsahoval z 20 $\cdot 10^6$ generací ($n_r = 1$, $n_c = 30$. Lepšího řešení lze dosáhnout, když přidáme další redundatní buňky. Doposud nejlepší známé řešení s 23 hradly bylo nalezeno s nastavením sítě $n_r = 1$, $n_c = 35$ [13]! Tento obvod se skládá ze 14 *AND* operátorů, 9 *XOR* operátorů a 2 *NOT* operátorů. Obvod je zobrazen na obrázku 4.3.



Obrázek 4.1: Nejlepší známá tříbitová násobička navržená konvenčním způsobem. Obsahuje 26 hradel. Převzato z [7].



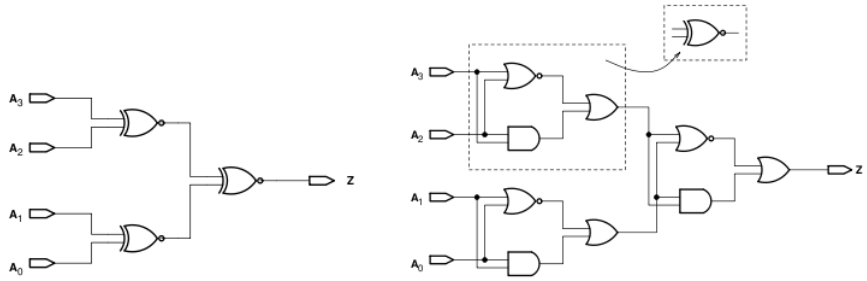
Obrázek 4.2: Tříbitová násobička, která obsahuje 24 dvouvstupých hradel. Je tak o 20 % lepší než konvenční tříbitová násobička z obrázku 4.1. Převzato z [7].



Obrázek 4.3: Nejlepší nalezená tříbitová násobička. Obsahuje pouze 23 dvouvstupých hradel. Je tak o více jak 20 % lepší než konvenční tříbitová násobička z obrázku 4.1. Převzato z [13].

4.1.2 Sudá parita

N-bitová sudá parita se stala standardním výkonnostním testem pro GP od chvíle, kdy ji Koza zmínil v jeho první knize [4]. Program nebo obvod pro výpočet paritního bitu vrací hodnotu *TRUE* (1) pokud je počet jedniček lichý. Pokud je počet sudý, obvod vrací *FALSE* (0) [3]. Aby bylo nalezení takového obvodu opravdu netriviálním problémem, je nutné, aby mezi množinou logických hradel nebyl XOR ani XNOR! Je to z důvodu, že například obvod pro čtyřbitovou sudou paritu lze sestavit pouze ze 3 hradel typu XNOR [7]. Příklad je vidět na obrázku 4.4.



Obrázek 4.4: Nalevo je vidět obvod pro čtyřbitovou paritní funkci sestavený pomocí hradel XNOR. Napravo je stejný obvod sestavený pouze z hradel AND, OR a NOR. Obrázek je převzatý z [7].

4.2 Statistické vyhodnocení

4.2.1 Úspěšnost nezávislých běhů

Pro vyšetření kvality experimentu slouží úspěšnost nezávistlých běhů, označme ji jako U. Tato hodnota říká, kolik procent nezávislých běhů dosáhlo řešení, tj. maximální hodnoty fitness. N_u je počet úspěšných běhů a N_{celkem} představuje celkový počet běhů. Pro výpočet slouží jednoduchá rovnice 4.1.

$$U = \frac{N_u}{N_{celkem}} \cdot 100. \tag{4.1}$$

4.2.2 Dosažené fitness hodnoty

Pro hlubší zkoumání výsledků z experimentů lze vyhodnocovat dosažené fitness hodnoty. Studium těchto hodnot může odhalit náhodné úspěchy[1], nebo zhodnotit měření, které nedosáhly úspěšného řešení.

Ke zpracování dosažených fitness hodnot může dobře posloužit zobrazení ve formě boxplotu. Boxplot obsahuje tyto hodnoty: minimální fitness, první kvartyl fitness hodnot, medián fitness hodnot, třetí kvartyl fitness hodnot a maximální fitness hodnotu.

[1] Ty se nejčastěji projevují jako velké výchylky od průměru/mediánu naměřených fitness hodnot.

4.2.3 Zhodnocení kvality výsledku a výpočtu

Z hlediska kvality řešení můžeme vyhodnocovat počet hradel a zpoždění obvodu T, potažmo cenu obvodu. Dalším důležitým měřítkem je celková doba výpočtu.

4.2.4 Computation effort

V [4] Koza popsal dnes již populární statistickou metodu "computation effort" (CE). Pomocí této metriky můžeme vypočítat minimální počet kandidátů, kteří musí být vyhodnoceni, aby byl bylo nalezeno řešení minimálně s pravděpodobností z, kde nejčastěji je volena hodnota $z = 99$ %. Tato metoda je vhodná k porovnávání výkonnosti různých variant GP především proto, že zohledňuje nejen úspěšnost nezávislých běhů měření, ale především zohledňuje, ve které generaci už dochází k nalezení řešení.

Pokud bychom měřili dvě různé metody na základě procentuální úspěšnosti nalezení řešení v nezávislých bězích, tak bychom v případě stejného procenta úspěšnosti nemohli porovnávat, která z metod je lepší. Přitom například první metoda by mohla nacházet většinu řešení už kolem 30. generace, ale druhá až kolem 100. generace. V případě maximálního omezení generací na 150 by to byl velký rozdíl.

Výpočet CE je zapsán v níže uvedených formulích. $N_u(i)$ je počet úspěchů[2] v nezávislých bězích do i. generace. N_{celkem} představuje celkový počet nezávislých běhů. $P(M, i)$ je kumulativní pravděpodobnost úspěchu pro nezávislý běh s populací o velikosti M dávající řešení do generace i. $R(P(M, i), z)$ je počet nezávislých běhů potřebných k nalezení řešení určené generací i s pravděpodobností z. $I(M, i, z)$ je počet kandidátů potřebných k nalezení řešení s pravděpodobností z s populací o velikosti M v generaci i [17].

$$P(M, i) = \frac{N_u(i)}{N_{celkem}}, \tag{4.2}$$

$$R(P(M, i), z) = \left\lceil \frac{log(1 - z)}{log(1 - P(M, i))}) \right\rceil, \tag{4.3}$$

$$I(M, i, z) = M \cdot R(P(M, i), z) \cdot (i + 1), \tag{4.4}$$

$$CE = \min_i I(M, i, z). \tag{4.5}$$

[2] tzn. bylo nalezeno alespoň jedno řešení

Kapitola 5

Navržené operátory křížení

V této kapitole jsou představeny tři navržené operátory křížení pro CGP. Operátory jsou inspirovány nebo odvozeny od standardních genetických operátorů, které byly představeny ve 2. kapitole. Všechny tři navržené operátory křížení jsou rozděleny do dvou různých variant. Varianty popisují způsob uplatnění zvoleného křížení a mutace na vybrané rodiče z minulé generace. Selekce rodičovských řešení je realizována podobně jako u stadardní varianty CGP, kde se vždy vybírá nejlepší jedinec[1]. Pro křížení jsou vždy vybráni dva nejlepší jedinci, přičemž nejlepší jedinec z minulé generace zůstává opět nezměněný, aby byl zachován princip algoritmu CGP pro prohledávání stavového prostoru, tj. ES $(1 + \lambda)$.

5.1 Vlastní implementace

Vlastní implementace kartézského genetického programování byla naprogramována v jazyce C++ s využitím standardních knihoven. Implementace je tedy multiplatformní. Testování probíhalo na operačním systému typu Linux.

5.1.1 Srovnání implementací

Implementace CGP ze sady Tools4CGP i vlastní implementace CGP obsahují stejnou sadu funkcí pro bloky, případně lze tuto sadu jednoduše rozšířit. Nastavení parametrů evoluce je u obou shodné. Vlastní implementace načítá při každém spouštění z textového souboru, kdežto CGP má na vstupu předzpracovaný .h soubor. Vlastní implementace, narozdíl od programu CGP ze sady Tools4CGP, neobsahuje realizaci paralelní simulace, která výrazně urychluje výpočet fitness. Z tohoto důvodu bylo upuštěno od implementace nových typu křížení do vlastní implementace CGP. Nové typy křížení byly implementovány do nástroje CGP ze sady Tools4CGP.

5.1.2 Modifikace implementace CGP ze sady Tools4CGP

Do veřejně dostupné implementace CGP ze sady Tools4CGP byly implementovány všechny operátory uvedené v této kapitole. Pro každé měření se zpracovává soubor se souhrnnými statistikami a soubor s nejlepším nalezeným chromozomem, který dosahuje maximální fitness hodnoty. Soubor se statistikou obsahuje nastavení parametrů evoluce a vybraný typ křížení. Pro následné statistické vyhodnocení obsahuje dosaženou úspěšnost pro všechny

[1] S vyjímkou, kdy se v rodičovském řešení vyskytuje takových řešení více. Potom je zvolen ten, který nebyl rodičem v předchozí generaci.

běhy (4.2.1), hodnoty pro vytvoření boxplotu (4.2.2), průměrnou fitness hodnotu a computational effort (4.2.4). Dále vypisuje počet hradel, jenž obsahovalo nejlepší nalezené řešení, které dosahuje maximální možné fitness hodnoty. Jako poslední je uvedena hodnota reálného času měření[2].

5.2 První varianta

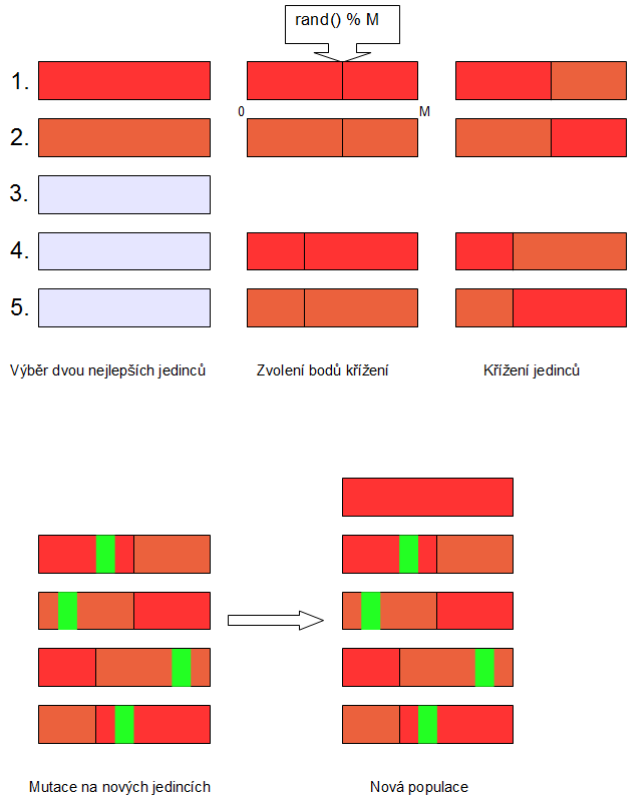
V této variantě křížení je na všechny nově vzniklé potomky z křížení aplikován operátor mutace. Délka chromozomu je v následujících popisech označena M.

5.2.1 Jednobodové křížení

Jednobodové křížení se dvakrát aplikuje na dva vybrané nejlepší rodiče z předchozí generace. Na celé délce chromozomu je náhodně vybrán bod křížení. Od tohoto bodu křížení se vymění zbylé části chromozomů. Toto křížení je popsáno jako algoritmus 3. Jeho grafické znázornění je na obrázku 5.1.

```
ALGORITMUS 3:
1b_krizeni(X, Y) {
    cross_index := random() % delka_chromozomu;
    for (i := 0; i < delka_chromozomu; i++) {
        if (i < cross_index) {
            X'[i] := X[i];
            Y'[i] := Y[i];
        } else {
            X'[i] := Y[i];
            Y'[i] := X[i];
        }
    }
    return X', Y';
}

X := zkopiruj_nejlepsi_chromozom();
Y := zkopiruj_druhy_nejlepsi_chromozom();
X1', Y1' := 1b_krizeni(X, Y);
X2', Y2' := 1b_krizeni(X, Y);
X1' := mutace(X1');
X2' := mutace(X2');
Y1' := mutace(Y1');
Y2' := mutace(Y2');
\\Nově vzniklá populace: X, X1', Y1', X2', Y2'
```

[2]V případě, že se procesy při výpočtu dělí o prostředky jednoho procesoru, nebo jsou spouštěny na různých výpočetních systémech, je tato hodnota silně nesignifikantní!

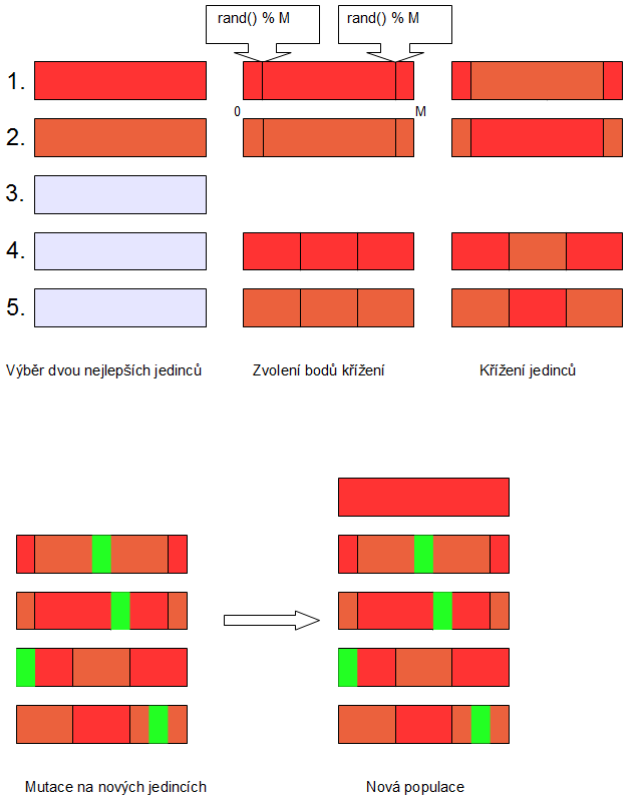


Obrázek 5.1: První varianta jednobodového křížení.

5.2.2 Dvoubodové křížení

Stejně tak, jako výše popsané jednobodové křížení, se i toto křížení aplikuje na dva vybrané nejlepší jedince z předchozí generace. Na celé délce chromozomu jsou náhodně vybrány 2 body křížení. Tyto body jsou opět reprezentovány jako celočíselné hodnoty. Od menšího bodu křížení po větší bod křížení se vymění části chromozomů. Ostatní geny jsou zkopírovány. Dvoubodové křížení je popsano jako algoritmus 4. Jeho grafické znázornění je na obrázku 5.2.

```
ALGORITMUS 4:
2b_krizeni(X, Y) {
    cross_index1 := random() % delka_chromozomu;
    cross_index2 := random() % delka_chromozomu;
    while(cross_index1 == cross_index2) {
        cross_index2 := random() % delka_chromozomu;
    }
    if (cross_index2 < cross_index1) {
        swap(cross_index1, cross_index2);
    }
    for (i := 0; i < delka_chromozomu; i++) {
        if (i < cross_index1 or i >= cross_index2) {
            X'[i] := X[i];
            Y'[i] := Y[i];
        } else {
            X'[i] := Y[i];
            Y'[i] := X[i];
        }
    }
    return X', Y';
}

X := zkopiruj_nejlepsi_chromozom();
Y := zkopiruj_druhy_nejlepsi_chromozom();
X1', Y1' := 2b_krizeni(X, Y);
X2', Y2' := 2b_krizeni(X, Y);
X1':= mutace(X1');
X2':= mutace(X2');
Y1':= mutace(Y1');
Y2':= mutace(Y2');
\\Nově vzniklá populace: X, X1', Y1', X2', Y2'
```

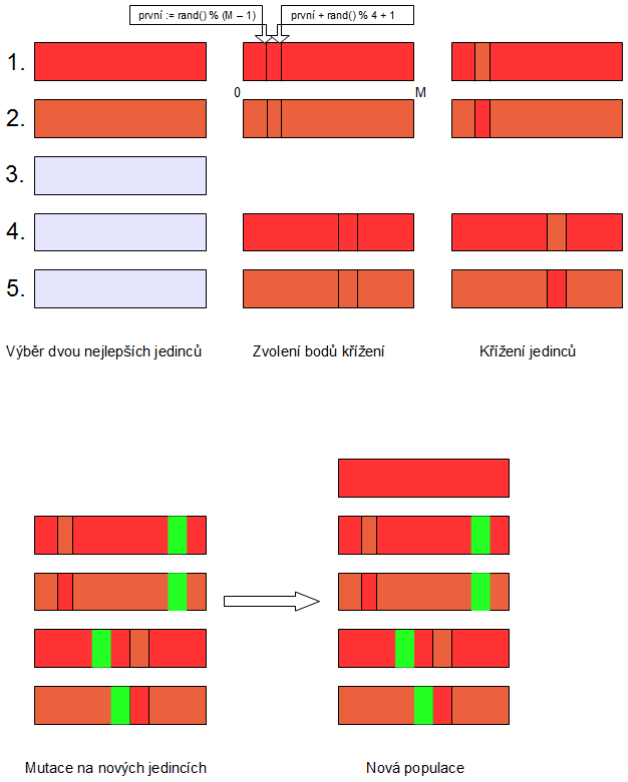


Obrázek 5.2: První varianta dvoubodového křížení.

5.2.3 Krátké dvoubodové křížení

Dvouvobodé krátké křížení je inspirováno motivací, proč se nepoužívá operátor křížení v CGP. Dvoubodové krátké křížení se vyhývá velkému rozbijení stavebních bloků, jak k tomu dochází u jednobodového a dvoubodového křížení. Toto křížení se od dvoubodového liší tím, že druhý bod je vybrán náhodně od prvního bodu křížení ve vzdálenosti maximálně 4 geny, jedná se zde o heuristiku. Podotkněme, že délka bloku v chromozomu, tj. počet celočíselných hodnot reprezentující jedno dvouvstupé hradlo, jsou 3 geny. Toto křížení je popsané jako algoritmus 5. Jeho grafické znázornění je na obrázku 5.3.

```
ALGORITMUS 5:
2b_kratke_krizeni(X, Y) {
    cross_index1 := random() % (delka_chromozomu - 1);
    cross_index2 := (random() % 4) + cross_index1 + 1;
    if (cross_index2 >= delka_chromozomu) {
        cross_index2 := delka_chromozomu - 1;
    }
    for (i := 0; i < delka_chromozomu; i++) {
        if (i < cross_index1 or i >= cross_index2) {
            X'[i] := X[i];
            Y'[i] := Y[i];
        } else {
            X'[i] := Y[i];
            Y'[i] := X[i];
        }
    }
    return X', Y';
}

X := zkopiruj_nejlepsi_chromozom();
Y := zkopiruj_druhy_nejlepsi_chromozom();
X1', Y1' := 2b_kratke_krizeni(X, Y);
X2', Y2' := 2b_kratke_krizeni(X, Y);
X1' := mutace(X1');
X2' := mutace(X2');
Y1' := mutace(Y1');
Y2' := mutace(Y2');
\\Nově vzniklá populace: X, X1', Y1', X2', Y2'
```



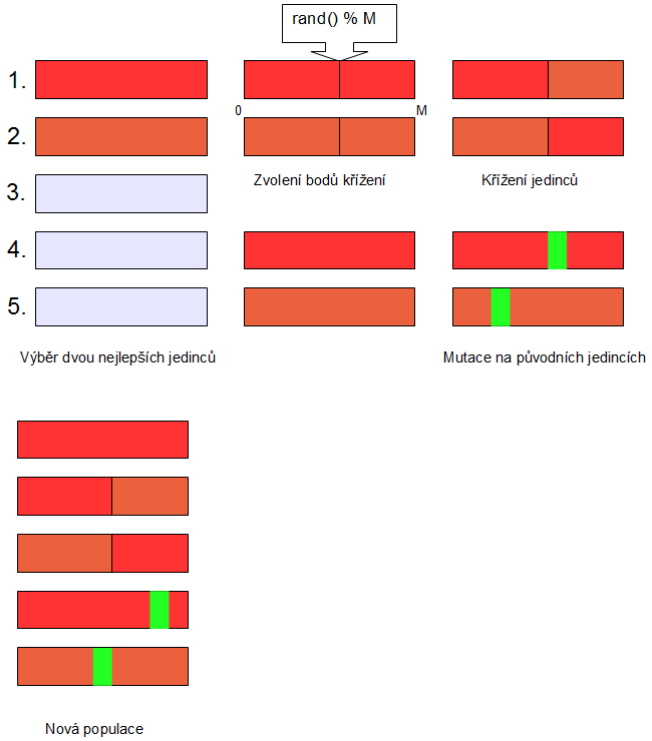
Obrázek 5.3: První varianta dvoubodového krátkého křížení.

5.3 Druhá varianta

Tato varianta uplatňuje operátor křížení pouze jedenkrát. Ostatní (dva) potomci vznikají pouze aplikací operátoru mutace na nejlepšího jedince z rodičovského řešení. To znamená, že vzniknou dva noví potomci křížením a dva potomci mutací.

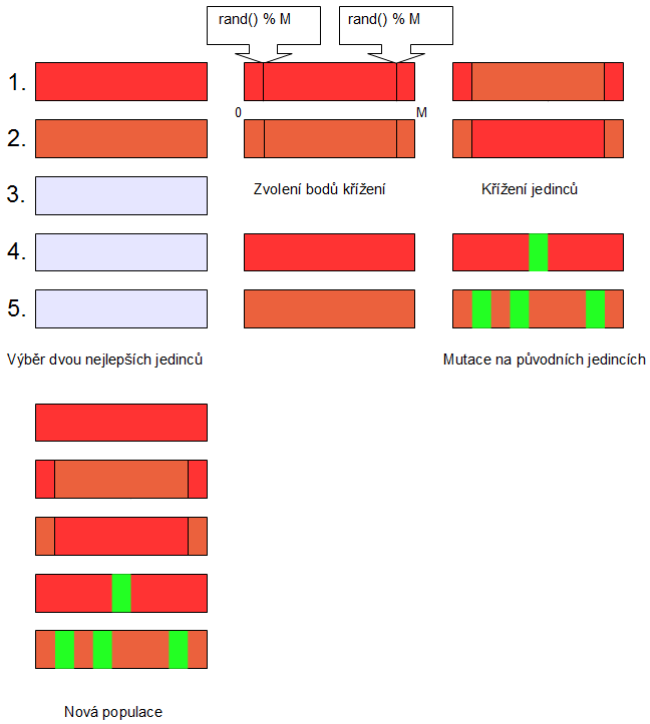
Jednobodové křížení (algoritmus 6) je graficky znázorněné na obrázku 5.4. Dvoubodové křížení (algoritmus 7) je graficky znázorněné na obrázku 5.5 . Dvoubodové krátké křížení (algoritmus 8) je vidět na obrázku 5.6.

```
ALGORITMUS 6:
1b_krizeni(X, Y) {
    cross_index := random() % delka_chromozomu;
    for (i := 0; i < delka_chromozomu; i++) {
        if (i < cross_index) {
            X'[i] := X[i];
            Y'[i] := Y[i];
        } else {
            X'[i] := Y[i];
            Y'[i] := X[i];
        }
    }
    return X', Y';
}

X := zkopiruj_nejlepsi_chromozom();
Y := zkopiruj_druhy_nejlepsi_chromozom();
X1', Y1' := 1b_krizeni(X, Y);
X2' := mutace(X');
Y2' := mutace(Y');
\\Nově vzniklá populace: X, X1', Y1', X2', Y2'
```



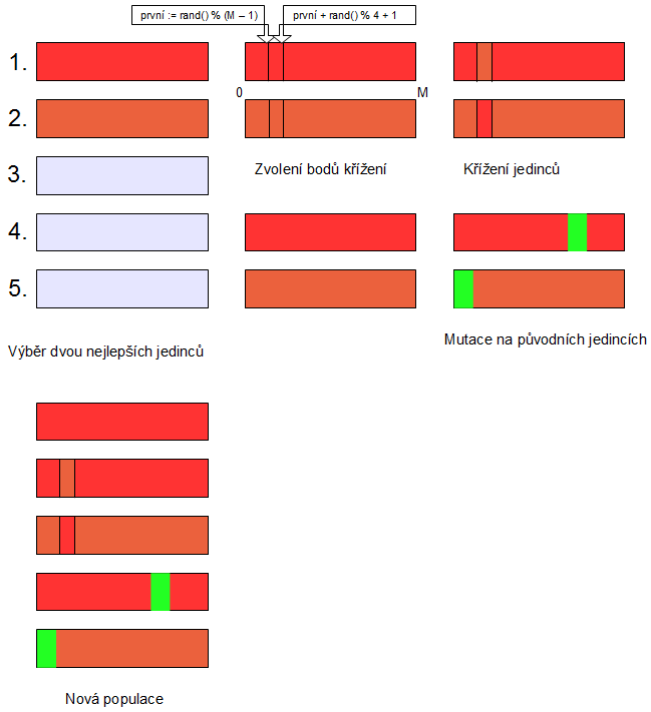
Obrázek 5.4: Druhá varianta jednobodového křížení.

```
ALGORITMUS 7:
2b_krizeni(X, Y) {
    cross_index1 := random() % delka_chromozomu;
    cross_index2 := random() % delka_chromozomu;
    while (cross_index1 == cross_index2) {
        cross_index2 := random() % delka_chromozomu;
    }
    if (cross_index2 < cross_index1) {
        swap(cross_index1, cross_index2);
    }
    for (i := 0; i < delka_chromozomu; i++) {
        if (i < cross_index1 or i >= cross_index2) {
            X'[i] := X[i];
            Y'[i] := Y[i];
        } else {
            X'[i] := Y[i];
            Y'[i] := X[i];
        }
    }
    return X', Y';
}

X := zkopiruj_nejlepsi_chromozom();
Y := zkopiruj_druhy_nejlepsi_chromozom();
X1', Y1' := 2b_krizeni(X, Y);
X2' := mutace(X');
Y2' := mutace(Y');
\\Nově vzniklá populace: X, X1', Y1', X2', Y2'
```



Obrázek 5.5: Druhá varianta dvoubodového křížení.

```
ALGORITMUS 8:
2b_kratke_krizeni(X, Y) {
    cross_index1 := random() % (delka_chromozomu - 1);
    cross_index2 := (random() % 4) + cross_index1 + 1;
    if (cross_index2 >= delka_chromozomu) {
        cross_index2 := delka_chromozomu - 1;
    }
    for (i := 0; i < delka_chromozomu; i++) {
        if (i < cross_index1 or i >= cross_index2) {
            X'[i] := X[i];
            Y'[i] := Y[i];
        } else {
            X'[i] := Y[i];
            Y'[i] := X[i];
        }
    }
    return X', Y';
}

X := zkopiruj_nejlepsi_chromozom();
Y := zkopiruj_druhy_nejlepsi_chromozom();
X1', Y1' := 2b_kratke_krizeni(X, Y);
X2' := mutace(X');
Y2' := mutace(Y');
\\Nově vzniklá populace: X, X1', Y1', X2', Y2'
```



Obrázek 5.6: Druhá varianta dvoubodového krátkého křížení.

Kapitola 6

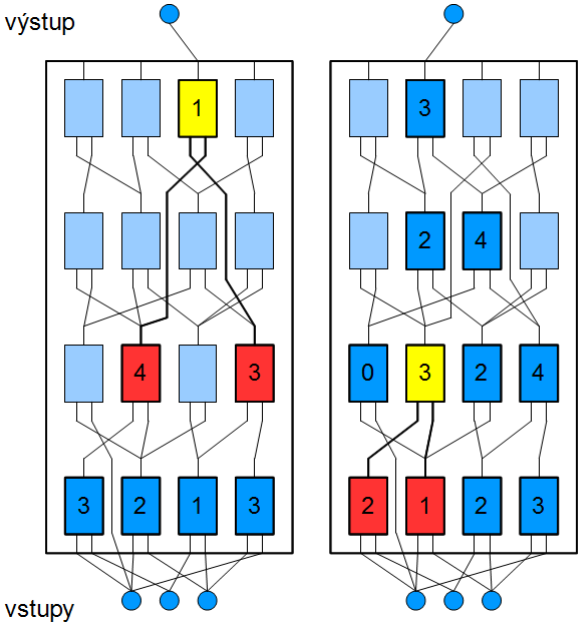
Navržené sémantické křížení

Navržené sémantické křížení je inspirováno sémantickým křížením pro GP, konkrétně variantou SSC z [11]. Hlavní myšlenkou tohoto křížení je křížit určité stavební celky, které mají velmi blízký sémantický význam.

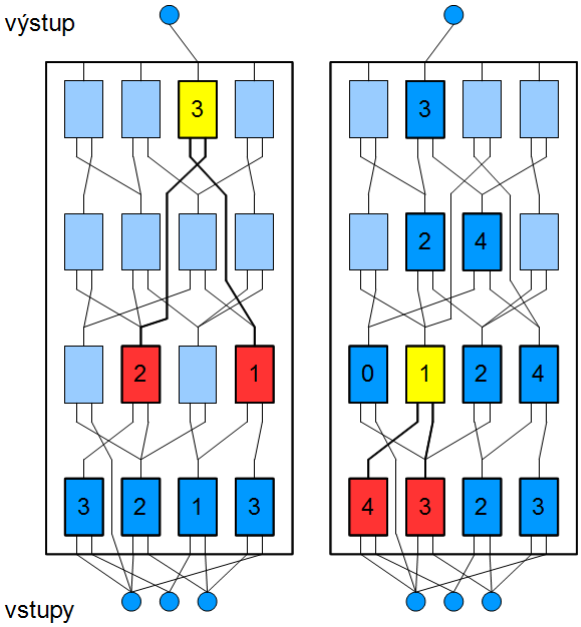
6.1 Popis křížení

Nejprve jsou pro oba rodičovské chromozomy nalezeny tzv. aktivní bloky zapojení. Aktivní bloky jsou bloky, které jsou použity v obvodu. V každém chromozomu je náhodně vybrán takový blok, který je aktivní, a vstupy do něj jsou z jiných bloků (ne z primárních vstupů). Takovému bloku budeme říkat hlavní. Pokud se hlavní bloky liší svou funkčností a odezva těchto obvodů, tj. hlavního bloku se dvěma vstupními bloky, je shodná ve více jak třech čtvrtinách, provede se křížení funkcí mezi hlavními bloky a mezi bloky vstupujícími do hlavních bloků. Pokud se náhodně vybrané hlavní bloky neliší nebo odezva podstromu je menší než tři čtvrtiny, hledají se nové hlavní bloky s novými vlastními podstromy. Celkový počet vstupů pro tento podstrom je 4. Tzn. pro výpočet odezvy je potřeba $2^4 = 16$ kombinací. Příklad výpočtu odezvy pro chromozomy uvedené na obrázku 6.1 je uvedený v tabulce 6.1. Jelikož je odezva u obou podstromů ve 12 vstupních kombinacích shodná, aplikuje se křížení. Výsledek po křížení je vidět v obrázku 6.2. Počet zkoušení hledání takových bloků je omezen pevně zadanou konstantou (MAX_POCET_ZKOUSENI). Na závěr se na nové chromozomy aplikuje operátor mutace. Metoda je popsaná jako algoritmus 9.

```
ALGORITMUS 9:
semanticke_krizeni(X, Y) {
    oznac_aktivni_bloky(X);
    oznac_aktivni_bloky(Y);
    pocet := 0;
    while (pocet < MAX_POCET_ZKOUSENI) {
        blok_x := nahodne_vyber_aktivni_blok(X);
        blok_y := nahodne_vyber_aktivni_blok(Y);
        fx := funkce_bloku(blok_x);
        fy := funkce_bloku(blok_y);
        if (fx != fy) {
            f1 := ziskej_prvni_funkci_vstupniho_bloku(blok_x);
            f2 := ziskej_druhou_funkci_vstupniho_bloku(blok_x);
            f3 := ziskej_prvni_funkci_vstupniho_bloku(blok_y);
            f4 := ziskej_druhou_funkci_vstupniho_bloku(blok_y);
            if (spocitej_podobnost(fx,f1,f2,fy,f3,f4) > 11) {
                swap(fx, fy);
                swap(f1, f3);
                swap(f2, f4);
                return X, Y;
            }
        pocet++;
        }
    }
    return X, Y;
}

X1', Y1' := semanticke_krizeni(X, Y);
X2', Y2' := semanticke_krizeni(X, Y);
X1' := mutace(X1');
X2' := mutace(X2');
Y1' := mutace(Y1');
Y2' := mutace(Y2');
\\Nově vzniklá populace: X, X1', Y1', X2', Y2'
```



Obrázek 6.1: Fenotyp před křížením prvního a druhého rodiče. Hlavní bloky jsou označeny žlutou barvou a vstupní červenou. Funkční blok F1 = NOT(in1 OR in2) a funkční blok F2 = (in1 XOR in2) AND NOT(in3). Γ = {NOT(in1) (1), XOR (2), AND (3), OR (4)}



Obrázek 6.2: Fenotyp po křížením prvního a druhého rodiče. Γ = {NOT(in1) (1), XOR (2), AND (3), OR (4)}

vstupy				**F1**			**F2**			**shoda**
in1	**in2**	**in3**	**in4**	**xor**	**not(in1)**	**and**	**or**	**and**	**not(in3)**	**F1=F2**
1	**1**	**1**	**1**	0	0	0	1	1	0	1
1	**1**	**1**	**0**	0	0	0	1	0	0	1
1	**1**	**0**	**1**	0	1	0	1	0	0	1
1	**1**	**0**	**0**	0	1	0	1	0	0	1
1	**0**	**1**	**1**	1	0	0	1	1	0	1
1	**0**	**1**	**0**	1	0	0	1	0	0	1
1	**0**	**0**	**1**	1	1	1	1	0	0	0
1	**0**	**0**	**0**	1	1	1	1	0	0	0
0	**1**	**1**	**1**	1	0	0	1	1	0	1
0	**1**	**1**	**0**	1	0	0	1	0	0	1
0	**1**	**0**	**1**	1	1	1	1	0	0	0
0	**1**	**0**	**0**	1	1	1	1	0	0	0
0	**0**	**1**	**1**	0	0	0	0	1	1	1
0	**0**	**1**	**0**	0	0	0	0	0	1	1
0	**0**	**0**	**1**	0	1	0	0	0	1	1
0	**0**	**0**	**0**	0	1	0	0	0	1	1
celkem:										12

Tabulka 6.1: Pravdivostni tabulka odezvy dvou podstromů. Shoda je nalezena pro 12 vstupních kombinací.

Kapitola 7

Experimentální ověření

V této kapitole jsou prezentovány a následně diskutovány výsledky měření pro navržené varianty křížení.

7.1 Přehled měření a metodika hodnocení

Větší část experimentů byla provedena pro hledání realizace tříbitové násobičky. Experimenty byly měřeny pro standardní variantu CGP. Výsledky těchto měření budou sloužit jako referenční vzorek pro srovnání kvality navržených operátorů křížení. Dále byly realizovány experimenty pro neinformované a sémantické křížení, které byly podrobně popsány v 5. a 6. kapitole. Protože navržené sémantické křížení jako jediné mělo signifikantně lepší výsledky v úspěšnosti běhů i pro computational effort než standardní varianta CGP bez křížení, jsou pro další srovnání této varianty uvedeny i výsledky měření v úloze pětibitové parity, a to v závěru kapitoly.

Hlavními ukazateli kvality navržených operátorů křížení byly zvoleny úspěšnost nezávislých běhů a computational effort. Jako vedlejší ukazatele slouží statistické vyhodnocení maximálně dosažených fitness hodnot nezávislých běhů. Pro úplnost jsou uvedena nejlepší nalezená řešení co do počtu použitých hradel. Zpoždění obvodu nebyla zpracována.

Pro lepší popisování konkrétních kombinací nastavených parametrů n_c a μ_g v následujících tabulkách byla zavedena jednoduchá notace $[n_c, \mu_g]$. Například notace [45, 2] ukazuje na hodnotu v tabulce s nastavením $n_c = 45$ a $\mu_g = 2$. Notace [30, 2-5] ukazuje na hodnoty s parametrem $n_c = 45$ a μ_g z množiny {2,3,4,5}.

Symbol "-"v následujích tabulkách znamená, že hodnota CE je pro dané měření neznámá, protože žádný z běhů nenalezl řešení.

7.2 Tříbitová násobička

Z výsledků měření popsaných ve 4. kapitole jasně vyplývá důležitost vlivu nastavení parametrů CGP. Pro experimenty s tříbitovou násobičkou použijeme nastavení doporučené v [6][8][13].

7.2.1 Nastavení parametrů

CGP nastavíme tak, aby byl prostor možných řešení co největší. Toho docílíme, když nastavíme $n_r = 1$ a $L = n_c$. Paramentr λ se obecně nastavuje na malé číslo. V tomto experimentu

nastavíme parametr $\lambda = 4$, což je nejobvyklejší hodnota. Tedy celková velikost populace $P = 5$. Maximální počet generací G omezíme na 20 milionů. Počáteční generace je generována náhodně. Množina dostupných hradel bude $\Gamma = \{AND, OR, XOR, NOT\}$. Výsledek jednoho běhu je relativně nezajímavý, může se jednat o náhodu. Proto bude provedeno až 50 běhů pro každé nastavení experimentu. Nastavovat budeme počet možných mutací a velikost rekonfigurovatelného obvodu. Rozsah mutace, tj. hodnota μ_g, budeme sledovat od 1 do 10. Velikost obvodu budeme ovlivňovat parametrem n_c s hodnotami 33, 35, 37, 39, 40 a 45. Celkový počet měření pro každé křížení je 10 (μ_g hodnot) $\times$ 6 (n_c hodnot) = 60. Počet evaluací bude u každého měření stejný, tj. součin $G \times P = 20 \cdot 10^6 \times 5 = 100 \cdot 10^6$. CE se bude počítat s hodnotou $z = 99$ %.

7.2.2 Standardní CGP

Největší úspěšnost nalezení řešení byly pro hodnoty v řádku $n_c = 45$. Z hlediska nastavení parametru mutace měl největší úspěch sloupec s parametrem $\mu_g = 2$. Největší dosažená úspěšnost byla 88 % právě na průsečíku těchto parametrů. Více viz tabulka 7.1.

Tabulka 7.2 popisuje kvalitu experimentu z hlediska hodnot CE. Zde byl suveréně nejlepší řádek s $n_c = 45$, ale řešení se nejrychleji nacházela s parametrem mutace $\mu_g = 1$. Nejlepší hodnotu CE má stejná kombinace [45, 2] jako v předešlé tabulce. Tato kombinace potřebuje k nalezení řešení ohodnotit minimálně 164 $\cdot 10^6$ kandidátních řešení, aby bylo nalezeno řešení s pravděpodobností 99 %.

Nejmenší počet hradel byl 27. Tento počet není mezi výsledky ojedinělý. Nastavení s nejlepší úspěšností nezávislých běhů [45, 2] i nastavení s nejlepším CE ohodnocením, tj. [45, 1], našlo řešení s počtem 27 hradel.

Nejlepší výsledky jsou dosaženy pro [45, 1-4]. V každé této kombinaci byly více jak 3/4 naměřených fitness hodnot rovny nebo větší než 383, přičemž maximální fitness je 384. Proto budeme tuto skupinu kombinací brát jako referenční vzorek pro porovnávání.

Důvodem nedosažení lepších vysledků, jaké byly popisovány v [13], je nespíš náhodná inicializace počáteční populace a poloviční počet běhů.

	μ_g									
n_c	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
33	22	14	16	12	16	8	10	8	0	6
35	34	44	24	18	10	22	20	10	14	8
37	52	46	36	24	28	26	20	26	14	10
39	62	62	58	52	36	38	24	30	36	20
40	66	70	62	54	44	50	50	34	20	20
45	80	88	84	82	80	66	66	70	54	60

Tabulka 7.1: Procentuální úspěšnost nalezení řešení pro standardní variantu CGP.

7.2.3 První varianta aplikace křížení

Jednobodové křížení

Tabulka 7.3 popisuje procentuální úspěšnost nalezení řešení v nezávislých běžích. Nejlepší úspěšnost 80 % i CE 238 $\cdot 10^6$ byla v [45, 3].

	μ_g									
n_c	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
33	1502	2872	1268	3428	651	4171	3781	5379	-	5517
35	1046	761	1391	1442	3240	1421	1983	2096	2025	4901
37	629	529	724	1254	1257	1130	2096	1514	2409	3563
39	456	364	513	430	842	738	1362	873	1004	1281
40	286	354	381	537	770	697	682	1010	1521	1711
45	173	164	181	166	281	317	382	332	588	410

Tabulka 7.2: Computational effort pro standardní variantu CGP. Hodnoty jsou uváděny v milionech.

	μ_g									
n_c	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
33	18	8	8	8	6	4	2	2	2	2
35	18	28	16	14	8	12	6	0	6	2
37	28	48	40	9	16	6	8	0	20	0
39	40	50	40	40	38	8	18	12	8	2
40	72	50	42	40	16	24	14	18	2	10
45	76	78	80	56	54	46	46	42	28	32

Tabulka 7.3: Procentuální úspěšnost nalezení řešení pro jednobodové křížení.

	μ_g									
n_c	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
33	2382	3904	5102	4370	55167	10586	16966	21901	6605	5820
35	1890	1394	1854	2627	4381	3565	5408	-	7372	19012
37	896	603	847	1908	1827	5955	4988	-	1980	-
39	410	665	932	938	946	5369	1584	3328	3236	16459
40	272	699	521	722	1873	1697	2594	1472	11245	4072
45	249	238	237	542	530	596	738	811	1244	1039

Tabulka 7.4: Computational effort pro jednobodové křížení. Hodnoty jsou uváděny v milionech.

Dvoubodové křížení

Popis procentuální úspěšnosti nalezení řešení v nezávislých bězích je v tabulce 7.5. Nejlepší úspěšnost 80 % i CE 235 $\cdot 10^6$ byla v kombinaci parametrů [45, 1].

	μ_g									
n_c	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
33	12	14	14	4	0	2	0	4	2	0
35	34	28	14	6	6	0	6	2	0	2
37	40	34	42	28	14	10	6	4	8	6
39	44	50	36	32	30	14	14	10	10	0
40	48	58	48	38	24	22	10	8	14	8
45	80	78	64	72	50	56	36	28	30	32

Tabulka 7.5: Procentuální úspěšnost nalezení řešení pro dvoubodové křížení.

	μ_g									
n_c	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
33	3634	2843	2082	10066	-	14231	-	10439	6809	-
35	818	1389	1867	4754	2317	-	5167	15442	-	20774
37	760	1110	778	1135	2228	2554	4363	10840	2104	5059
39	594	637	1027	1185	1075	2340	3077	3066	3792	-
40	453	566	682	668	1299	1341	2945	3479	2327	3366
45	235	267	433	327	497	444	669	1440	1099	1143

Tabulka 7.6: Computational effort pro dvoubodové křížení. Hodnoty jsou uváděny v milionech.

Krátké dvoubodové křížení

Tabulka 7.7 popisuje procentuální úspěšnost nalezení řešení v nezávislých bězích pro krátké dvoubodové křížení. Nejlepší úspěšnost byla opět v [45, 1] a měla až 90 %. Z tabulky 7.8 lze na stejné kombinaci parametrů vyčíst nejlepší hodnotu CE = 198 $\cdot 10^6$!

	μ_g									
n_c	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
33	22	10	8	4	8	4	4	6	2	0
35	32	28	14	16	8	8	6	6	2	0
37	46	30	26	16	14	14	14	4	18	6
39	54	42	34	40	28	14	8	28	10	14
40	58	46	42	30	32	24	26	16	26	20
45	90	70	78	64	60	68	50	28	34	26

Tabulka 7.7: Procentuální úspěšnost nalezení řešení pro dvoubodové krátké křížení.

	μ_g									
n_c	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
33	623	1753	4439	3830	5281	7617	11281	7200	19602	-
35	476	1478	2817	2153	2729	5274	2378	4169	21351	-
37	578	968	1390	1915	2529	1540	1642	4018	2159	7225
39	341	375	826	881	1079	2473	1430	1360	3824	3046
40	444	409	857	960	982	1486	1381	2023	1275	1770
45	198	288	311	333	538	421	541	1197	1057	1563

Tabulka 7.8: Computational effort pro dvoubodové krátké křížení. Hodnoty jsou uváděny v milionech.

7.2.4 Druhá varianta aplikace křížení

Jednobodové křížení

Tabulka 7.9 popisuje procentuální úspěšnost nalezení řešení v nezávislých bězích pro druhou variantu jednobodového křížení. Nejlepší úspěšnost byla opět v [45, 1], ale měla pouze 74 %. Z tabulky 7.10 lze na stejné kombinaci parametrů vyčíst nejlepší hodnotu CE = 207 $\cdot 10^6$.

	μ_g									
n_c	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
33	8	8	6	0	6	2	4	2	0	0
35	14	12	12	16	8	8	2	2	0	2
37	36	30	16	24	10	8	4	4	2	6
39	46	28	28	22	16	16	8	8	8	2
40	38	44	30	14	26	22	8	6	8	8
45	74	56	56	56	50	36	32	32	24	14

Tabulka 7.9: Procentuální úspěšnost nalezení řešení pro druhou variantu jednobodového křížení.

	μ_g									
n_c	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
33	4411	4343	7276	-	6225	9287	9577	13907	-	-
35	2199	2503	3338	2691	4302	4691	18574	18282	-	4033
37	1068	1219	2674	1611	2630	5234	8720	6859	8443	5064
39	661	1198	1238	1684	1544	952	4988	2371	3331	18154
40	714	789	1179	1874	1306	1246	4339	6476	3185	5079
45	207	394	428	527	682	910	1047	1023	1375	2260

Tabulka 7.10: Computational effort pro druhou variantu jednobodového křížení. Hodnoty jsou uváděny v milionech.

Dvoubodové křížení

Tabulka 7.11 popisuje procentuální úspěšnost nalezení řešení v nezávislých bězích pro druhou variantu dvoubodového křížení. Nejlepší změřená úspěšnost byla pouze 62 % a nejlepší hodnota CE = 434 $\cdot 10^6$.

	μ_g									
n_c	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
33	10	2	2	2	0	0	2	2	0	0
35	18	5	12	8	4	6	2	8	2	4
37	24	32	18	20	16	12	10	2	2	0
39	34	30	22	22	26	10	14	12	14	4
40	36	44	24	32	18	10	4	10	14	0
45	50	52	56	62	34	42	32	30	16	24

Tabulka 7.11: Procentuální úspěšnost nalezení řešení pro druhou variantu dvoubodového křížení.

	μ_g									
n_c	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
33	3562	17231	8287	17707	-	-	21045	22005	-	-
35	2055	1799	3450	5291	9471	6023	14705	5563	13844	11232
37	729	979	2312	1736	2119	2939	2717	11122	13941	-
39	868	1058	1848	1415	1269	3416	2880	1462	2780	8298
40	634	508	1571	1142	1982	3267	10950	4245	2365	-
45	548	451	560	434	959	808	1058	1215	2117	1671

Tabulka 7.12: Computational effort pro druhou variantu dvoubodového křížení. Hodnoty jsou uváděny v milionech.

Krátké dvoubodové křížení

Tabulka 7.13 popisuje procentuální úspěšnost nalezení řešení v nezávislých bězích pro druhou variantu dvoubodového krátkého křížení. Nejlepší změřená úspěšnost byla pouze 54 % a nejlepší hodnota CE = 442 $\cdot 10^6$.

	μ_g									
n_c	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
33	8	4	2	4	0	0	0	0	0	0
35	8	8	2	2	0	0	0	0	0	0
37	18	10	6	0	2	2	0	0	2	0
39	32	14	14	8	6	2	2	0	0	0
40	30	30	16	8	8	2	2	2	2	4
45	54	44	38	26	14	14	10	10	8	6

Tabulka 7.13: Procentuální úspěšnost nalezení řešení pro druhou variantu dvoubodového krátkého křížení.

	μ_g									
n_c	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
33	2594	10928	18010	9921	-	-	-	-	-	-
35	3358	4000	14928	16311	-	-	-	-	-	-
37	1633	4086	6526	-	21092	15726	-	-	22607	-
39	1025	2373	2291	4015	7005	13013	6746	-	-	-
40	1249	1275	2241	3870	4473	5319	15932	10255	12742	8300
45	442	643	892	1472	2645	2725	4395	4135	4912	5677

Tabulka 7.14: Computational effort pro druhou variantu dvoubodového krátkého křížení. Hodnoty jsou uváděny v milionech.

7.2.5 Vyhodnocení první a druhé varianty

Nejlepší úspěšnost řešení byla vždy nalezena v rozmezí [45, 1-4]. Žebříček pořadí jednotlivých křížení na základě průměrné hodnoty v těchto rozmezí je uveden v tabulce 7.15. Pořádí jednotlivý křížení jsou stejná jak pro úspěšnost nalezení řešení, tak i pro CE. Výsledný žebříček potvrdil předpoklad nefunkčnosti standardních variant křížení pro CGP. Druhé pořadí první varianty dvoubodového křížení nám dává signál, že omezením křížení na kratší část chromozomu nemusí tolik degenerovat populaci, jak se to děje při rozbijení důležetých bloků u ostatních variant křížení v CGP.

pořadí	název křížení	průměr úsp. z [45, 1-4]	průměr CE z [45, 1-4]
1.	standardní CGP	83.5 %	171 $\cdot 10^6$
2.	I. varianta 2b krátké křížení	75.5 %	283 $\cdot 10^6$
3.	I. varianta 2b křížení	73.5 %	315 $\cdot 10^6$
4.	I. varianta 1b křížení	72.5 %	316 $\cdot 10^6$
5.	II. varianta 1b křížení	60.5 %	389 $\cdot 10^6$
6.	II. varianta 2b křížení	55.0 %	499 $\cdot 10^6$
7.	II. varianta 2b krátké křížení	40.5 %	862 $\cdot 10^6$

Tabulka 7.15: Žebříček hodnocení standardní varianty CGP a křížením s I. nebo s II. variantami.

7.2.6 Sémantické křížení

Pro sémantické křížení se měřilo pouze pro parametr $n_c = 45$ a $\mu_g = \{1, 2, 3, 4\}$. Koeficient počtu hledání vhodné kombinace pro křížení MT[1] $= \{5, 10, 15\}$.

	μ_g			
MT	**1**	**2**	**3**	**4**
5	96	94	86	84
10	96	92	84	80
15	92	92	90	72

Tabulka 7.16: Procentuální úspěšnost nalezení řešení pro sémantické křížení.

	μ_g			
MT	**1**	**2**	**3**	**4**
5	102	141	234	257
10	153	175	219	293
15	152	154	182	326

Tabulka 7.17: CE pro sémantické křížení. Hodnoty jsou uváděny v milionech.

[1] Zkratka je odvozena z konstanty MAX_TRIAL uvedené v pseudokódu SSC z [11]

7.2.7 Vyhodnocení sémantického křížení

Procentuální úspěšnosti uvedené v tabulce 7.16 jsou ve srovnání se standardním CGP ve většině případů mnohem lepší. Hodnoty CE uvedené v tabulce 7.17 jsou jen pro některé hodnoty parametru MT lepší než ve standardním CGP. Průměrná úspěšnost pro [45, 1-4] s MT = 5 je 90 %! Průměrně dosažený CE pro [45, 1-4] je 184 $\cdot 10^6$. To znamená, že se řešení nachází trochu později, ale s mnohem větší pravděpodobností.

Za zmínku dále stojí, že jak úspěšnost tak i CE měly pro kombinace MT = 5 s $\mu_g = 1$ doposud nejlepší výsledky z celého měření! Průměrná fitnes byla 383.959991. Počet nalezených hradel byl v této kombinaci parametrů 28. Nejmenší počet nalezených hradel v celém měření byl 27.

Z pohledu doby měření představuje sémantické křížení, oproti standardní variantě CGP, řežii navíc. Čas trvání standardní varianty CGP pro jeden běh s 10 $\cdot 10^6$ generacemi byl přibližně 218 sekund. Čas trvání CGP se sémantickým křížením pro stejné nastavení s MT=5 trvalo až 563 sekund.

7.3 Pětibitová sudá parita

7.3.1 Nastavení parametrů

Paramentry opět nastavíme tak, aby byl prostor možných řešení co největší. Nastavíme $n_r = 1$ a $L = n_c$. Celková velikost populace bude opět $P = 5$. Maximální počet generací G omezíme na $70 \cdot 10^3$. Počáteční generace je generována náhodně. Množina dostupných hradel nebude obsahovat XOR ani XNOR, $\Gamma = \{AND, OR, NOT\}$. Bude provedeno 50 běhů pro každé nastavení experimentu. U standardní varianty CGP budeme opět sledovat výsledky různý počet možných mutací a velikost obvodu. Rozsah mutace, tj. hodnota μ_g, budeme sledovat v rozmezí od 1 do 5. Velikost obvodu budeme opět ovlivňovat pouze parametrem n_c s hodnotami 25, 30 a 35. Celkový počet měření pro každé křížení je 5 (μ_g hodnot) $\times$ 3 (n_c hodnot) = 15. U varianty se sémantickým křížením budeme zkoumat MT pro hodnoty 5, 10 a 15. Počet evaluací bude u každého měření stejný, tj. součin $G \times P = 70 \cdot 10^3 \times 5 = 350 \cdot 10^3$.

7.3.2 Standardní CGP

Nejvyšší hodnota úspěšnosti 82 % byla změřena pro [35, 5]. Pro stejnou kombinaci byla spočítána nejlepší hodnota CE $9 \cdot 10^6$. Jako nejlepší se jeví výsledky s parametrem $n_c = 35$. Průměr hodnot dosažených úspěšností nezávislých běhů v této variantě je 65 %. Průměrný CE $= 16 \cdot 10^6$. Více viz tabulka 7.18 a 7.19. Všechny měřené kombinace parametrů nacházely řešení, které měly 16 hradel.

	μ_g				
n_c	**1**	**2**	**3**	**4**	**5**
25	24	24	24	24	24
30	48	50	48	34	50
35	52	60	68	64	82

Tabulka 7.18: Procentuální úspěšnost nalezení řešení pro standardní CGP.

	μ_g				
n_c	**1**	**2**	**3**	**4**	**5**
25	49	49	55	49	58
30	26	23	26	40	24
35	22	17	15	16	9

Tabulka 7.19: Hodnoty CE pro standardní CGP. Hodnoty jsou uváděny v milionech.

7.3.3 Sémantické křížení

Tabulky 7.20, 7.22 a 7.24 popisují úspěšnost nezávislých běhů pro různé nastavení parametru n_c. Tabulky 7.21, 7.23 a 7.25 popisují hodnoty CE pro různé nastavení parametru n_c.

	μ_g				
MT	**1**	**2**	**3**	**4**	**5**
5	16	20	12	20	6
10	24	30	18	14	12
15	18	14	18	10	18

Tabulka 7.20: Procentuální úspěšnost nalezení řešení pro sémantické křížení s $n_c = 25$.

	μ_g				
MT	**1**	**2**	**3**	**4**	**5**
5	91	61	122	68	216
10	58	43	81	100	97
15	83	89	78	151	79

Tabulka 7.21: Hodnoty CE pro sémantické křížení s $n_c = 25$. Hodnoty jsou uváděny v milionech.

	μ_g				
max. počet pokusů	**1**	**2**	**3**	**4**	**5**
5	34	34	40	40	30
10	24	26	30	34	28
15	30	42	28	26	36

Tabulka 7.22: Procentuální úspěšnost nalezení řešení pro sémantické křížení s $n_c = 30$.

	μ_g				
MT	**1**	**2**	**3**	**4**	**5**
5	39	38	31	33	45
10	52	54	43	39	51
15	45	31	50	45	35

Tabulka 7.23: Hodnoty CE pro sémantické křížení s $n_c = 30$. Hodnoty jsou uváděny v milionech.

	μ_g				
MT	**1**	**2**	**3**	**4**	**5**
5	58	60	58	66	50
10	40	52	68	56	56
15	48	50	44	50	42

Tabulka 7.24: Procentuální úspěšnost nalezení řešení pro sémantické křížení s $n_c = 35$.

	μ_g				
MT	**1**	**2**	**3**	**4**	**5**
5	20	17	19	16	22
10	28	23	14	16	20
15	25	23	27	22	29

Tabulka 7.25: Hodnoty CE pro sémantické křížení s $n_c = 35$. Hodnoty jsou uváděny v milionech.

7.3.4 Vyhodnocení sémantického křížení

Nejlepší dosažená úspěšnost ze všech možných nastavení byla pouhých 68 %. Nejlepší CE byl $16 \cdot 10^6$. Nejlepší variantou se jeví nastavení s parametrem $n_c = 35$ a MT $= 5$. Průměr hodnot dasažených úspěšností nezávislých běhů v této variantě je necelých 59 %. Průměrný CE $= 19 \cdot 10^6$. Všechny měřené varianty nacházely řešení, které měly 16 hradel.

7.4 Zhodnocení výsledků a realizace měření

Výsledky měření pro hledání realizace pětibitové sudé parity neměly tak dobré výsledky, jako výsledky pro hledání tříbitové násobičky. Naměřené hodnoty byly dokonce horší, než pro referenční standardní variantu CGP. Tím se nepotvrdila robustnost navrhnutého sémantického křížení.

Kapitola 8

Závěr

Tato diplomová práce v teoretické části zpracovává základní informace o CGP a seznamuje tak s nejnovějšími trendy a poznatky v oblasti návrhu číslicových obvodů pomocí CGP. Byl popsán obecný princip evolučních algoritmů a biologií inspirovaných genetických operátorů. V další kapitole je představen princip CGP. Nemalá část je v této kapitole věnovaná popisu implementace CGP ze sady Tools4CGP.

Hlavní část této práce se zabývá křížením v CGP. V 5. kapitole jsou navrženy nové operátory křížení, které jsou inspirovány standardními operátory křížení. V následující kapitole je navrženo sémantické křížení. Tyto křížení byly implementovány a testovány na dvou zvolených problémech.

Výsledky experimentů potvrdily předpoklad nevhodnosti standardních operátorů křížení pro CGP. Slibněji dopadly výsledky měření pro navržené sémantické křížení. Pro hledání obvodu tříbitové násobičky mělo sémantické křížení s vhodným nastavením 90% úspěšnost nalezení řešení. Standardní varianta CGP měla pouze 83% úspěšnost nalezení řešení v nezávislých bězích. Jako druhý testovací problém byla zvolena pětibitová parita. Výsledky pro hledání tohoto obvodu nepotvrdily předešlé kvality navrženého sémantického křížení. Příčin může být několik. Na vině může být špatně zvolený nízký parametr počtu bloků, nebo nevhodnost použití tohoto operátoru na malé obvody.

Pro lepší vyhodnocení sémantického křížení je potřeba provést další měření i pro jiné obvody. V rámci této diplomové práce nebyl už na tato další měření prostor.

Literatura

[1] Clegg, J.; Walker, J. A.; Miller, J. F.: A new crossover technique for Cartesian genetic programming. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, London: ACM Press, 2007, s. 1580–1587.

[2] Drábek, V.: *Výstavba počítačů*. Skriptum VUT Brno, 1995.

[3] Harding, S.; Miller, J. F.; Banzhaf, W.: Self modifying cartesian genetic programming: Parity. In *Cartesian Genetic Programming*, Natural Computing Series, Springer Berlin Heidelberg, 2009, s. 285–292.

[4] Koza, J. R.: *Genetic programming - on the programming of computers by means of natural selection*. Complex adaptive systems, MIT Press, 1993, ISBN 978-0-262-11170-6.

[5] Miller, J.; Thomson, P.: *Cartesian Genetic Programming*. LNCS, Springer Verlag, 2000, s. 121–132.

[6] Miller, J. F.: Cartesian Genetic Programming. In *Cartesian Genetic Programming*, editace J. F. Miller, Natural Computing Series, Springer Berlin Heidelberg, 2011, ISBN 978-3-642-17310-3, s. 17–34.

[7] Miller, J. F.; Job, D.; Vassilev, V. K.: Principles in the Evolutionary Design of Digital Circuits - Part I. *Genetic Programming and Evolvable Machines*, ročník 1, April 2000: s. 7–35, ISSN 1389-2576.

[8] Sekanina, L.: *Evoluční hardware: od automatického generování patentovatelných invencí k sebemodifikujícím se strojům*. ACADEMIA, 2009, ISBN 978-80-200-1729-1.

[9] Sekanina, L.; Walker, A. J.; Kaufmann, P.; aj.: Evolution of Electronic Circuits. In *Cartesian Genetic Programming*, editace J. F. Miller, Natural Computing Series, Springer Berlin Heidelberg, 2011, ISBN 978-3-642-17310-3, s. 125–179.

[10] Slaný, K.; Sekanina, L.: Fitness landscape analysis and image filter evolution using functional-level CGP. In *Proceedings of the 10th European conference on Genetic programming*, EuroGP'07, Springer-Verlag, 2007, ISBN 978-3-540-71602-0, s. 311–320.

[11] Uy, N. Q.; Hoai, N. X.; O'Neill, M.; aj.: Semantically-based crossover in genetic programming: application to real-valued symbolic regression. *Genetic Programming and Evolvable Machines*, ročník 12, č. 2, 2011: s. 91–119.

[12] Vasicek, Z.; Sekanina, L.: Formal verification of candidate solutions for post-synthesis evolutionary optimization in evolvable hardware. *Genetic Programming and Evolvable Machines*, ročník 12, 2011: s. 305–327, ISSN 1389-2576, 10.1007/s10710-011-9132-7.

[13] Vassilev, V. K.; Job, D.; Miller, J.: Towards the Automatic Design Of More Efficient Digital Circuits. In *Proc. of the 2nd NASA/DoD Workshop on Evolvable Hardware*, IEEE Computer Society, 2000, s. 151–160.

[14] Vašíček, Z.: Evoluční návrh kombinačních obvodů zadaných tabulkou. Technická zpráva, Vysoké učení technické v Brně, 2004.

[15] Vašíček, Z.; Sekanina, L.: Evoluční návrh kombinačních obvodů. *Elektrorevue - www.elektrorevue.cz*, ročník 2004, č. 43, 2004: s. 1–6, ISSN 1213-1539.
URL http://www.fit.vutbr.cz/research/view_pub.php?id=7539

[16] Vašíček Zdeněk, S. L.: Nástroje pro kartézské genetické programování. Technická zpráva, Vysoké učení technické v Brně, 2008.
URL http://www.fit.vutbr.cz/research/prod/index.php?id=61

[17] Walker, J. A.; Miller, J. F.; Kaufmann, P.; aj.: Problem Decomposition in Cartesian Genetic Programming. In *Cartesian Genetic Programming*, editace J. F. Miller, Natural Computing Series, Springer Berlin Heidelberg, 2011, s. 35–99.

Příloha A

Obsah DVD

- písemná zpráva
- zdrojové kódy písemné zprávy
- zdrojové kódy modifikace CGP ze sady Tools4CGP
- soubory se souhrnými statistikami z experimentálních měření